

UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTTTTTTTTTTTTTTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEE	TTT	PPP	PPP
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEEEEEEEEEEEEEEE	TTT	PPPPPPPPPPPPPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUU	UUU	EEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	
UUUUUUUUUUUUUUUU	UUUUUUUUUUUUUUUU	EEEEEEEEEEEEEEEE	TTT	PPP	

UU	UU	EEEEEEEEEE	TTTTTTTTTT	DDDDDDDD	MM	MM	PPPPPPPP	FFFFFFFFFF	000000	000000	
UU	UU	EEEEEEEEEE	TTTTTTTTTT	DDDDDDDD	MM	MM	PPPPPPPP	FFFFFFFFFF	000000	000000	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EEEEEEEEEE	TT	DD	DD	MM	PPPPPPPP	FFFFFFFFFF	00	0000	
UU	UU	EEEEEEEEEE	TT	DD	DD	MM	PPPPPPPP	FFFFFFFFFF	00	0000	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UU	UU	EE	TT	DD	DD	MM	PP	FF	00	00	
UUUUUUUUUU	EEEEEEEEEE	TT	DDDDDDDD	MM	MM	PP	PP	FF	000000	000000	
UUUUUUUUUU	EEEEEEEEEE	TT	DDDDDDDD	MM	MM	PP	PP	FF	000000	000000	

LL	IIIIII	SSSSSSSS
LL	IIIIII	SSSSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SSSSSS
LL	II	SSSSSS
LL	II	SS
LL	II	SS
LL	II	SS
LL	II	SS
LLLLLLLLLL	IIIIII	SSSSSSSS
LLLLLLLLLL	IIIIII	SSSSSSSS



(2)	86	Declarations
(3)	164	Read-Only Data
(4)	301	Read/Write Data
(5)	489	RMS-32 Data Structures
(6)	543	Main Program
(11)	852	Test the DMP/DMF
(12)	1151	CHECKIOSB - Check IO status block
(13)	1212	Check QIO AST Routine
(14)	1253	Receive data AST routine
(15)	1310	Half Minute Timer Expiration Routine
(16)	1352	Three Minutes Timer Expiration Routine
(18)	1387	System Service Exception Handler
(19)	1516	RMS Error Handler
(20)	1580	CTRL/C Handler
(21)	1631	Error Exit
(22)	1697	Exit Handler



```
0000 1 .TITLE UETDMPF00 VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF-32 Sync Line
0000 2 .IDENT 'V04-001'
0000 3 :
0000 4 :*****
0000 5 :
0000 6 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 :* ALL RIGHTS RESERVED.
0000 9 :
0000 10 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 :* TRANSFERRED.
0000 16 :
0000 17 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 :* CORPORATION.
0000 20 :
0000 21 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 :
0000 24 :*
0000 25 :*****
0000 26 :
0000 27 :
0000 28 :++
0000 29 : FACILITY:
0000 30 : This module will be distributed with VAX/VMS under the [SYSTEST]
0000 31 : account.
0000 32 :
0000 33 : ABSTRACT:
0000 34 : This is the test program for DMP 11 / DMF 32 sync line device test
0000 35 :
0000 36 : ENVIRONMENT:
0000 37 : This program will run in user access mode, with AST enabled except
0000 38 : during error processing. This program requires the following
0000 39 : privileges and quotas: none.
0000 40 :
0000 41 :--
0000 42 :
0000 43 : AUTHOR: Paul Jenq, CREATION DATE: Sep, 1981
0000 44 :
0000 45 : MODIFIED BY:
0000 46 :
0000 47 : V04-001 RNH0009 Richard N. Holstein, 07-Sep-1984
0000 48 : Remove entirely the forced error for too big a buffer, since
0000 49 : either SS$_BADPARAM or SS$_EXQUOTA could be returned depending
0000 50 : on SYSGEN parameters.
0000 51 :
0000 52 : V03-010 RNH0008 Richard N. Holstein, 07-Apr-1984
0000 53 : Adapt to driver fix which allocated write buffers dynamically -
0000 54 : we can't force an SS$_BADPARAM unless we exceed absolute max.
0000 55 :
0000 56 : V03-009 RNH0007 Richard N. Holstein, 15-Feb-1984
0000 57 : Take advantage of the new UETP message codes. Fix SSERROR
```



```

0000 58 : interaction with RMS_ERROR.
0000 59 :
0000 60 : V03-008 RNH0006 Richard N. Holstein, 19-Dec-1983
0000 61 : Give correct sentinels to Test Controller.
0000 62 :
0000 63 : V03-007 RNH0005 Richard N. Holstein, 11-Nov-1983
0000 64 : Use decimal conversion routine for unit numbers.
0000 65 :
0000 66 : V03-006 RNH0004 Richard N. Holstein, 29-Jun-1983
0000 67 : Rework error messages and error processing.
0000 68 :
0000 69 : V03-005 RNH0003 Richard N. Holstein, 11-Mar-1983
0000 70 : Don't signal ending message in EXIT_HANDLER.
0000 71 :
0000 72 : V03-004 RNH0002 Richard N. Holstein, 01-Mar-1983
0000 73 : Fix ERROR_COUNT bug.
0000 74 :
0000 75 : V03-003 LDJ0002 Larry D. Jones, 10-Feb-1983
0000 76 : Allow for longer device names.
0000 77 :
0000 78 : V03-002 LDJ0001 Larry D. Jones, 06-Nov-1982
0000 79 : Fixed a loop mode assign channel bug.
0000 80 :
0000 81 : V03-001 RNH0001 Richard N. Holstein, 15-Oct-1982
0000 82 : Miscellaneous fixes listed in the V3B UETP Workplan.
0000 83 :
0000 84 : **

```



```
0000 86      .SBTTL Declarations
0000 87      :
0000 88      : INCLUDE FILES:
0000 89      :
0000 90      :      SYSS$LIBRARY:LIB.MLB      for general definitions
0000 91      :      SHRLIB$:UETP.MLB          for UETP definitions
0000 92      :
0000 93      :
0000 94      : MACROS:
0000 95      :
0000 96      :      $CHFDEF                      ; Condition handler frame definitions
0000 97      :      $DEVDEF                      ; Device definitions
0000 98      :      $DIBDEF                      ; Device Information Block
0000 99      :      $DVIDEF                     ; $GETDVI ITMLST item codes
0000 100     :      $SHRDEF                      ; Shared messages
0000 101     :      $SSDEF                       ; System Service status codes
0000 102     :      $STSDEF                      ; Status return
0000 103     :      $UETUNTDEF                   ; UETP unit block offset definitions
0000 104     :      $UETPDEF                     ; UETP
0000 105     :      $XMDEF                       ; XMDRIVER symbols
0000 106     :      $NMADEF                      ; Network management definition
0000 107     :
0000 108     : EQUATED SYMBOLS:
0000 109     :
0000 110     : Facility number definitions:
0000 111     :      RMS$_FACILITY = 1
0000 112     :
0000 113     : SHR message definitions:
00740000 0000 114     :      UETP = UETP$_FACILITY@STSSV FAC_NO ; Define the UETP facility code
007410E0 0000 115     :      UETP$_ABENDDD = UETP!SHR$_ABENDDD ; Define the UETP message codes
00741038 0000 116     :      UETP$_BEGINDD = UETP!SHR$_BEGINDD
00741080 0000 117     :      UETP$_ENDEDDE = UETP!SHR$_ENDEDDE
00741098 0000 118     :      UETP$_OPENIN = UETP!SHR$_OPENIN
00741130 0000 119     :      UETP$_TEXT = UETP!SHR$_TEXT
0000 120     :
0000 121     : Internal flag bits...:
00000001 0000 122     :      TEST_OVERV = 1 ; Set when test is over
00000002 0000 123     :      SAFE_TO_UPDV = 2 ; Set if it's safe to update UETINIDEV
00000003 0000 124     :      BEGIN_MSGV = 3 ; Set if "BEGIN" msg has been printed
00000004 0000 125     :      MODE_IS_ONEV = 4 ; Set when the MODE is ONE
00000005 0000 126     :      FLAG_SHOTDNV = 5 ; Set to indicate device should be
0000 127     :      ; shutdown if errors occur
0000 128     : ...and corresponding masks:
00000002 0000 129     :      TEST_OVERM = 1@TEST_OVERV
00000004 0000 130     :      SAFE_TO_UPDM = 1@SAFE_TO_UPDV
00000008 0000 131     :      BEGIN_MSGM = 1@BEGIN_MSGV
00000010 0000 132     :      MODE_IS_ONEM = 1@MODE_IS_ONEV
00000020 0000 133     :      FLAG_SHOTDNM = 1@FLAG_SHOTDNV
0000 134     :
0000 135     : Miscellany:
00000020 0000 136     :      LC_BITM = ^X20 ; Mask to convert lower case to upper
00000028 0000 137     :      REC_SIZE = 40 ; UETINIDEV.DAT record size
000000FA 0000 138     :      TEXT_BUFFER = 250 ; Internal text buffer size
00000004 0000 139     :      EFN2 = 4 ; EFN used for three minute timer
00000003 0000 140     :      SS_SYNCH_EFN = 3 ; Synch miscellaneous system services
0000000F 0000 141     :      MAX_PROC_NAME = 15 ; Longest possible process name
0000000A 0000 142     :      MAX_DEV_DESIG = 10 ; Longest possible controller name
```



```
00000005 0000 143 MAX_UNIT_DESIG= 5 ; Longest possible unit number
00000200 0000 144 MAX_MSG_LEN = 512 ; maximum message length
00000001 0000 145 TIME_ID_1 = 1 ; Timer id to prevent hung
00000003 0000 146 RW_TIME_ID = 3 ; Timer to prevent hung when Read/write
00000010 0000 147 LIMIT = 16 ; Loop count for each message length
00000008 0000 148 RECV_EFN = 8 ; EFN for QIO write
00000005 0000 149 XMIT_EFN = 5 ; AST parameter for test
00000064 0000 150 PRM = 100 ; Size of device dependent part of UETUNT
00000000 0000 151 DEVDEP_SIZE = 0 ; Size of device write buffer
00000000 0000 152 WRITE_SIZE = 0 ; Size of device read buffer
00000000 0000 153 READ_SIZE = 0
00000000 0000 154
00000000 0000 155 PAGES = <<UETUNT$C_INDSIZ+- ; Add together all of the pieces...
00000000 0000 156 DEVDEP_SIZE+- ; ...which make up a UETP unit block...
00000000 0000 157 WRITE_SIZE+- ; ...to give to the $EXPREG service below
00000000 0000 158 READ_SIZE+-
00000001 0000 159 511>7512>
0000001B 0000 160
00000004 0000 161 ESC = ^X1B ; ESC character
00000004 0000 162 RECVPOOL_SIZE = 4 ; Number of preallocated message block
```



```
0000 164 .SBTTL Read-Only Data
0000 165 .PSECT RODATA,NOEXE,NOWRT,PAGE
0000 166
53 45 54 53 59 53 00000008'010E0000' 0000 167 ACNT_NAME: ; Process name on exit
54 000E 168 .ASCID /SYSTEST/
000F 169
50 4D 44 54 45 55 00000017'010E0000' 000F 170 TEST_NAME: ; This test name
30 30 46 001D 171 .ASCID /UETDMPF00/
0020 172
50 55 53 54 45 55 00000028'010E0000' 0020 173 SUPDEV_GBLSEC: ; How we access UETSUPDEV.DAT
56 45 44 002E 174 .ASCID /UETSUPDEV/
0031 175
41 4E 4C 52 54 43 00000039'010E0000' 0031 176 CONTROLLER: ; Logical name of controller
45 4D 003F 177 .ASCID /CTRLNAME/
0041 178
45 44 4F 4D 00000049'010E0000' 0041 179 MODE: ; Run mode logical name
004D 180 .ASCID /MODE/
004D 181
00000000' 004D 182 NO_RMS_AST_TABLE: ; List of errors for which...
00000000' 0051 183 .LONG RMSS_BLN ; ...RMS cannot deliver an AST...
00000000' 0055 184 .LONG RMSS_BUSY ; ...even if one has an ERR= arg
00000000' 0059 185 .LONG RMSS_CDA ; Note that we can search table...
00000000' 005D 186 .LONG RMSS_FAB ; ...via MATCHC since <31:16>...
00000014 0061 187 .LONG RMSS_RAB ; ...pattern can't be in <15:0>
0061 188 NRAT_LENGTH = .-NO_RMS_AST_TABLE
0061 189
4E 49 24 53 59 53 00000069'010E0000' 0061 190 SYSS$INPUT: ; Name of device from which...
54 55 50 006F 191 .ASCID /SYSS$INPUT/ ; ...the test can be aborted
0072 192
0020 0040 0072 193 INPUT_ITMLST: ; $GETDVI arg list for SYSS$INPUT
0000000C'00000014' 0076 194 .WORD 64,DVIS$ DEVNAM ; We need the equivalence name
00000000 007E 195 .LONG BUFFER,BUFFER_PTR
0082 196 .LONG 0 ; Terminate the list
0082 197
21 20 42 58 32 21 0000008A'010E0000' 0082 198 CS1: ; Device class and type control string
20 42 58 32 0090 199 .ASCID /!2XB !2XB /
0094 200
2A 20 42 58 32 21 0000009C'010E0000' 0094 201 CS3: ; Device class-only control string
2A 0094 202 .ASCID /!2XB **/
00A2 203
00A3 204 CNTRLMSG:
00A3 205 .ASCID \Aborted via a user CTRL/C\
00B1 206
43 2F 4C 52 54 43 20 00BD 207 NO_CTRLNAME:
00C4 208 .ASCID /No controller specified./
00C4 209
6E 6F 63 20 6F 4E 000000CC'010E0000' 00C4
63 65 70 73 20 72 65 6C 6C 6F 72 74 00D2
2E 64 65 69 66 69 00DE
00E4
```



```
20 74 27 6E 61 43 000000EC'010E0000' 00E4 210 DEAD_CTRLNAME:
6C 6F 72 74 6E 6F 63 20 74 73 65 74 00E4 211 .ASCID /Can't test controller !AS, marked as unusable in UETINIDEV.DAT./
72 61 6D 20 2C 53 41 21 20 72 65 6C 00F2
61 73 75 6E 75 20 73 61 20 64 65 6B 00FE
4E 49 54 45 55 20 6E 69 20 65 6C 62 010A
2E 54 41 44 2E 56 45 44 49 0116
0122
012B 212
012B 213 NOUNIT_SELECTED:
012B 214 .ASCID /No units selected for testing./
0139
0145
0151 215
0151 216 ILLEGAL_REC:
0151 217 .ASCID /Illegal record format in file UETINIDEV.DAT!/
015F
016B
0177
0183
0185
0185 218
0185 219 PASS_MSG:
0193 220 .ASCID /End of pass !UL with !UL iterations at !%D./
019F
01AB
01B7
01B8
01B8 221
01B8 222 INIDEV_UPDERR: ; Error during exit handler
01C6 223 .ASCID /Error updating UETINIDEV.DAT./
01D2
01DD 224
01DD 225 THREEMIN: ; 3 minute delta time
01DD 226 .LONG -10*1000*1000*180,-1
01E5 227
01E5 228 HALFMIN: ; 30 seconds delta time
01E5 229 .LONG -10*1000*1000*30,-1
01ED 230
01ED 231 UNIT_DESC: ; Descriptor used to convert unit #
01ED 232 .LONG 5
01F1 233 .ADDRESS BUFFER+6
01F5 234
01F5 235 CONT_DESC: ; Descriptor used to convert controller...
01F5 236 .WORD REC_SIZE,0 ; ...from lowercase to uppcase
01F9 237 .ADDRESS BUFFER
01FD 238
01FD 239 FILE: ; Fills in RMS_ERR_STRING
01FD 240 .ASCID /file/
0209 241
0209 242 RECORD: ; Fills in RMS_ERR_STRING
0209 243 .ASCID /record/
0217 244
0217 245 RMS_ERR_STRING: ; Announces an RMS error
0217 246 .ASCID /RMS !AS error in file !AD/
0225
0231
0238 247
```



```
64 20 72 65 6C 6C 6F 72 74 6E 6F 43 0238
3A 3F 6E 6F 69 74 61 6E 67 69 73 65 0238
                                20 0244
                                00000019 0250
                                0251
                                0251
                                0251
75 6F 65 6D 69 54 00000259'010E0000' 0251
20 6F 74 20 67 6E 69 79 72 74 20 74 025F
    2E 53 41 21 20 74 72 61 74 73 026B
                                0275
                                0275
75 6F 65 6D 69 54 0000027D'010E0000' 0275
64 61 65 72 20 65 6C 69 68 77 20 74 0283
69 74 69 72 77 20 72 6F 20 67 6E 69 028F
    2E 53 41 21 20 67 6E 029B
                                02A2
                                02A2
20 72 6F 72 72 45 000002AA'010E0000' 02A2
20 70 75 20 67 6E 69 74 72 61 74 73 02B0
6E 6F 63 20 61 20 73 61 20 53 41 21 02BC
    2E 72 65 6C 6C 6F 72 74 02C8
                                02D0
                                02D0
20 72 6F 72 72 45 000002D8'010E0000' 02D0
20 70 75 20 67 6E 69 74 72 61 74 73 02DE
69 72 74 20 61 20 73 61 20 53 41 21 02EA
    2E 79 72 61 74 75 62 02F6
                                02FD
                                02FD
20 72 6F 72 72 45 00000305'010E0000' 02FD
21 20 6F 74 20 67 6E 69 74 69 72 77 030B
    2E 53 41 0317
                                031A
                                031A
20 72 6F 72 72 45 00000322'010E0000' 031A
6D 6F 72 66 20 67 6E 69 64 61 65 72 0328
    2E 53 41 21 20 0334
                                0339
                                0339
20 72 6F 72 72 45 00000341'010E0000' 0339
72 61 68 63 20 67 6E 69 73 6E 65 73 0347
20 73 63 69 74 73 69 72 65 74 63 61 0353
    2E 53 41 21 20 66 6F 035F
                                0366
                                0366
20 72 6F 72 72 45 0000036E'010E0000' 0366
72 61 68 63 20 67 6E 69 74 74 65 73 0374
20 73 63 69 74 73 69 72 65 74 63 61 0380
    2E 53 41 21 20 66 6F 038C
                                0393
                                0393
74 73 20 4F 2F 49 0000039B'010E0000' 0393
63 20 6B 63 6F 6C 62 20 73 75 74 61 03A1
74 73 20 6E 6F 69 74 65 6C 70 6D 6F 03AD
74 20 2C 57 58 21 20 3A 73 75 74 61 03B9
65 7A 69 73 20 72 65 66 73 6E 61 72 03C5
```

```
248 PROMPT:
249      .ASCII /Controller designation?: /

250      PMTSIZ = .-PROMPT
251
252 START_TO_MSG:
253      .ASCID /Timeout trying to start !AS./

254
255 RW_TO_MSG:
256      .ASCID /Timeout while reading or writing !AS./

257
258 START_CONT_PRM:
259      .ASCID /Error starting up !AS as a controller./

260
261 START_TRIB_PRM:
262      .ASCID /Error starting up !AS as a tributary./

263
264 WRITE_PRM:
265      .ASCID /Error writing to !AS./

266
267 READ_PRM:
268      .ASCID /Error reading from !AS./

269
270 SENSE_PRM:
271      .ASCID /Error sensing characteristics of !AS./

272
273 SET_PRM:
274      .ASCID /Error setting characteristics of !AS./

275
276 DMF_IOSB_DUMP:
277      .ASCID \I/O status block completion status: !XW, transfer size: !XW,\-
```



```
65 74 63 61 72 61 58 63 5F 21 2F 21 03D1
42 58 21 20 3A 73 63 69 74 73 69 72 03D7 278 \!/_characteristics: !XB, status: !XB, error summary: !XB.\
58 21 20 3A 73 75 74 61 74 73 20 2C 03E3
6D 75 73 20 72 6F 72 72 65 20 2C 42 03EF
2E 42 58 21 20 3A 79 72 61 6D 03FB
0407
0411 279
0411 280 DMP_IOSB_DUMP:
281 .ASCID \I/O status block completion status: !XW, transfer size: !XW,\-
74 73 20 4F 2F 49 00000419'010E0000' 0411
63 20 6B 63 6F 6C 62 20 73 75 74 61 041F
74 73 20 6E 6F 69 74 65 6C 70 6D 6F 042B
74 20 2C 57 58 21 20 3A 73 75 74 61 0437
65 7A 69 73 20 72 65 66 73 6E 61 72 0443
2C 57 58 21 20 3A 044F
65 74 63 61 72 61 68 63 5F 21 2F 21 0455 282 \!/_characteristics: !XB, status: !XB, error summary: !XB,\-
42 58 21 20 3A 73 63 69 74 73 69 72 0461
58 21 20 3A 73 75 74 61 74 73 20 2C 046D
6D 75 73 20 72 6F 72 72 65 20 2C 42 0479
2C 42 58 21 20 3A 79 72 61 6D 0485
75 6E 20 6C 61 74 6F 74 5F 21 2F 21 048F 283 \!/_total number of errors: !XB.\
6F 72 72 65 20 66 6F 20 72 65 62 6D 049B
2E 42 58 21 20 3A 73 72 04A7
04AF 284
04AF 285 COMP_STATUS_MSG:
286 .ASCID /Failure during forced error tests,/<13><10><9>/expected: ''/
72 75 6C 69 61 46 000004B7'010E0000' 04AF
72 6F 66 20 67 6E 69 72 75 64 20 65 04BD
65 74 20 72 6F 72 72 65 20 64 65 63 04C9
63 65 70 78 65 09 0A 0D 2C 73 74 73 04D5
22 20 3A 64 65 74 04E1
04E7 287
04E7 288 RECEIVED_MSG:
289 .ASCID /',/<13><10><9>/received: ''/
72 09 0A 0D 2C 22 000004EF'010E0000' 04E7
22 20 3A 64 65 76 69 65 63 65 04F5
04FF 290
04FF 291 RECV_ERR_MSG:
292 .ASCID /Received message error, good data is !XB, bad data is !XB /
76 69 65 63 65 52 00000507'010E0000' 04FF
65 20 65 67 61 73 73 65 6D 20 64 65 050D
64 20 64 6F 6F 67 20 2C 72 6F 72 72 0519
20 2C 42 58 21 20 73 69 20 61 74 61 0525
20 73 69 20 61 74 61 64 20 64 61 62 0531
20 42 58 21 053D
0541 293
0541 294 SENSE_ERRMSG:
295 .ASCID \Error in sense mode test, extended characteristic parameter\-\
20 72 6F 72 72 45 00000549'010E0000' 0541
64 6F 6D 20 65 73 6E 65 73 20 6E 69 054F
65 74 78 65 20 2C 74 73 65 74 20 65 055B
74 63 61 72 61 68 63 20 64 65 64 6E 0567
61 72 61 70 20 63 69 74 73 69 72 65 0573
72 65 74 65 6D 057F
68 74 69 77 20 57 58 21 5F 21 2F 21 0584 296 \!/_!XW with value !XL not matched by any of those returned.\
6E 20 4C 58 21 20 65 75 6C 61 76 20 0590
62 20 64 65 68 63 74 61 6D 20 74 6F 059C
6F 68 74 20 66 6F 20 79 6E 61 20 79 05A8
2E 64 65 6E 72 75 74 65 72 20 65 73 05B4
05C0 297
05C0 298 ERRTEST_MSG:
299 .ASCID /Error in error test /
20 72 6F 72 72 45 000005C8'010E0000' 05C0
```



UETDMPF00  
V04-001

VAX/VMS UETP DEVICE TEST FOR DMP <sup>I</sup>1/<sup>1</sup> DMF  
Read-Only Data

16-SEP-1984 01:24:05  
10-SEP-1984 12:03:55

VAX/VMS Macro V04-00  
[UETP.SRC]UETDMPF00.MAR;2

Page 9  
(3)

73 65 74 20 72 6F 72 72 65 20 6E 69 05CE  
20 74 05DA

UET  
V04



```
05DC 301 .SBTTL Read/Write Data
00000000 302 .PSECT RWDATA,WRT,NOEXE,PAGE
0000 303
0000 304 TTCHAN: ; Channel associated with ctrl. term.
0000 305 .WORD 0
0002 306
0002 307 FLAG: ; Miscellaneous flag bits
0000 308 .WORD 0 ; (See Equated Symbols for definitions)
0004 309
0004 310 FAO_BUF: ; FAO output string descriptor
0000 00FA 311 .WORD TEXT_BUFFER,0
00000014 312 .ADDRESS BUFFER
000C 313
000C 314 BUFFER_PTR: ; Fake .ASCID buffer for misc. strings
0000 00FA 315 .WORD TEXT_BUFFER,0 ; A word for length, a word for desc.
00000014 316 .ADDRESS BUFFER
0014 317
0014 318 BUFFER: ; FAO output and other misc. buffer
0000010E 319 .BLKB TEXT_BUFFER
010E 320
010E 321 ALT_FAO_BUF: ; FAO output string descriptor...
0000 00FA 322 .WORD TEXT_BUFFER,0 ; ...during ASTs
0000011E 323 .ADDRESS ALT_BUFFER
0112 324
0116 325 ALT_BUFFER_PTR: ; Fake .ASCID buffer for misc. strings
0000 00FA 326 .WORD TEXT_BUFFER,0 ; A word for length, a word for desc.
0000011E 327 .ADDRESS ALT_BUFFER ; Used during ASTs
011E 328
011E 329 ALT_BUFFER: ; FAO output and other misc. buffer...
00000218 330 .BLKB TEXT_BUFFER ; ...during ASTs
0218 331
0218 332 DEVDESC: ; Device name descriptor
0000 000A 333 .WORD MAX_DEV_DESIG,0
00000237 334 .ADDRESS DEV_NAME
0220 335
0220 336 PROCESS_NAME: ; Process name
46 50 4D 44 00000228 010E0000 337 .ASCID /DMPF/
022C 338
00000008 022C 339 PROCESS_NAME_FREE = MAX_PROC_NAME-<.-8-PROCESS_NAME>
00000237 022C 340 .BLKB PROCESS_NAME_FREE
0237 341
0237 342 DEV_NAME: ; Device name buffer
00000246 0237 343 .BLKB MAX_DEV_DESIG+MAX_UNIT_DESIG
0000000F 0246 344 NAME_LEN = .-DEV_NAME
0246 345
0246 346 DIB: ; Device Information Block
0000 0074 0246 347 .WORD DIB$K_LENGTH,0
0000024E 024A 348 .ADDRESS DIBBUF
024E 349
000002C2 024E 350 .BLKB DIB$K_LENGTH
02C2 351
02C2 352 ERROR_COUNT: ; Cumulative error count at runtime
00000000 02C2 353 .LONG 0
02C6 354
02C6 355 STATUS: ; Status value on program exit
00000000 02C6 356 .LONG 0
02CA 357
```



00000000	00000000	02CA	358	QUAD_STATUS:			; IO status block for misc sys. svcs.
		02CA	359	.QUAD	0		
		02D2	360				
		02D2	361	INADDRESS:			; \$CRMPSC address storage
00000000	00000000	02D2	362	.LONG	0,0		
		02DA	363	OUTADDRESS:			
00000000	00000000	02DA	364	.LONG	0,0		
		02E2	365				
		02E2	366	UNIT_NUMBER:			; Current dev unit number
	0000	02E2	367	.WORD	0		
		02E4	368				
		02E4	369	DEVNAM_LEN:			; Current device name length
	0000	02E4	370	.WORD	0		
		02E6	371				
		02E6	372	ITERATION:			; # of times all tests were executed
00000000		02E6	373	.LONG	0		
		02EA	374				
		02EA	375	PASS:			; Pass count
00000000		02EA	376	.LONG	0		
		02EE	377				
		02EE	378	MSG_BLOCK:			; Auxiliary \$GETMSG info
000002F2		02EE	379	.BLKB	4		
		02F2	380				
		02F2	381	EXIT_DESC:			; Exit handler descriptor
00000000		02F2	382	.LONG	0		
00000C26		02F6	383	.ADDRESS	EXIT_HANDLER		
00000001		02FA	384	.LONG	1		
000002C6		02FE	385	.ADDRESS	STATUS		
		0302	386				
		0302	387	ARG_COUNT:			; Argument counter used by ERROR_EXIT
00000000		0302	388	.LONG	0		
		0306	389				
		0306	390	XD_CHAN:			; DMP/F circuit channel
	0000	0306	391	.WORD	0		
		0308	392				
		0308	393	BUF_LEN:			; Length of primary chars
	0000	0308	394	.WORD	0		
		030A	395				
		030A	396	BUF_DESC:			; Get channel char buffer descriptor
00000074		030A	397	.LONG	DIB\$K_LENGTH		
00000312		030E	398	.LONG	CHAN_BUF		
		0312	399				
		0312	400	CHAN_BUF:			; Channel char buffer
00000386		0312	401	.BLKB	DIB\$K_LENGTH		
		0386	402				
		0386	403	P1BUF:			; P1 Device char buffer
00000000	00000000	0386	404	.QUAD	0		
		038E	405				
		038E	406	TR_P1BUF:			; p1 buufer for trib
00000000	00000000	038E	407	.QUAD	0		
		0396	408				
		0396	409	P2BUF_DESC:			; P2 extended char buffer
0000000C		0396	410	.LONG	P2BUF_LEN		
0000039E		039A	411	.ADDRESS	P2BUF		
		039E	412				
		039E	413	P2BUF:			; P2 extended buffer
0458		039E	414	.WORD	NMASC_PCLI_PRO		; Protocol mode



```
00000000 03A0 415 .LONG NMASC_LINPR_POI ; DDCMP print-to-point mode
03A4 416
0456 03A4 417 .WORD NMASC_PCLI_CON ; Controller mode
00000001 03A6 418 .LONG NMASC_LINCR_LOO ; Loopback mode
03AA 419
0000000C 03AA 420 P2BUF_LEN = .-P2BUF
03AA 421
03AA 422 TR_P2BUF_DESC: ; P2 extended char buffer for trib
00000006 03AA 423 .LONG TR_P2BUF_LEN
000003B2 03AE 424 .ADDRESS TR_P2BUF
03B2 425
03B2 426 TR_P2BUF: ; P2 extended buffer for trib
0474 03B2 427 .WORD NMASC_PCCI_TRI ; tributary address
00000001 03B4 428 .LONG 1 ; Address
03B8 429
00000006 03B8 430 TR_P2BUF_LEN = .-TR_P2BUF
03B8 431
03B8 432 SENSE_P1BUF: ; P1 buffer for sense mode test
00000000 00000000 03B8 433 .QUAD 0
03C0 434
03C0 435 SENSE_P2DESC: ; P2 buffer descrip for sense mode test
00000090 03C0 436 .LONG SENSE_P2LEN
000003C8 03C4 437 .ADDRESS SENSE_P2BUF
03C8 438
03C8 439 SENSE_P2BUF: ; P2 buffer for sense mode test
00000458 03C8 440 .BLKW <3*24> ; 8 quad quad words for dev information
00000090 0458 441 SENSE_P2LEN = .-SENSE_P2BUF ; P2 buffer length
0458 442
0458 443 ERRST_P2DESC: ; P2 desc for error test
00000008 0458 444 .LONG ERRST_P2LEN
00000460 045C 445 .ADDRESS ERRST_P2BUF
0460 446
0460 447 ERRST_P2BUF: ; P2 buffer for error test
00000468 0460 448 .BLKW 1
00000008 0468 449 ERRST_P2LEN = .-ERRST_P2BUF
0468 450
0468 451 ERRCOUNT_DESC: ; Error counter buffer descrip
00000200 0468 452 .LONG ERRCNT_LEN
00000470 046C 453 .ADDRESS ERRCNT_BUF
0470 454
0470 455 ERRCNT_BUF: ; Buffer for error counters
00000670 0470 456 .BLKW 64 ; Buffer length
00000200 0670 457 ERRCNT_LEN = .-ERRCNT_BUF
0670 458
0670 459 XD_IOSB: ; Q10 IO status block for transmit
00000678 0670 460 .BLKW 1
0678 461
0678 462 RCV_IOSB: ; Q10 IO status block for receive
00000680 0678 463 .BLKW 1
0680 464
0680 465 XMIT_BUF: ; Transmit buffer
00000880 0680 466 .BLKB MAX_MSG_LEN
0880 467
0880 468 RECV_BUF: ; Receive buffer
00000A80 0880 469 .BLKB MAX_MSG_LEN
0A80 470
0A80 471 BAD_DATA: ; Received wrong data
```



```

00 0A80 472 .BYTE 0
0A81 473
0A81 474 GOOD_DATA: ; Data sent (good)
00 0A81 475 .BYTE 0
0A82 476
0A82 477
0A82 478 ;
0A82 479 ; Head of self-relative UETP unit block queue.
0A82 480 ;
0A82 481 .ALIGN QUAD
0A88 482
0A88 483 UNIT_LIST: ; Head of unit block circular list
00000000 00000000 0A88 484 .QUAD 0
0A90 485
0A90 486 NEW_NODE: ; Newly acquired node address
00000000 00000000 0A90 487 .QUAD 0

```



```
0A98 489 .SBTTL RMS-32 Data Structures
0A98 490 .ALIGN LONG
0A98 491
0A98 492 SYSIN_FAB: ; Allocate FAB for SYSS$INPUT
0A98 493 $FAB-
0A98 494 FNM = <SYSS$INPUT>
0AE8 495
0AE8 496 SYSIN_RAB: ; Allocate RAB for SYSS$INPUT
0AE8 497 $RAB-
0AE8 498 FAB = SYSIN_FAB,-
0AE8 499 ROP = PMT,-
0AE8 500 PBF = PROMPT,-
0AE8 501 PSZ = PMTSIZ,-
0AE8 502 UBF = DEV_NAME,-
0AE8 503 USZ = NAME_LEN
0B2C 504
0B2C 505 INI_FAB: ; Allocate FAB for UETINIDEV
0B2C 506 $FAB-
0B2C 507 FAC = <GET,PUT,UPD>,-
0B2C 508 RAT = CR,-
0B2C 509 SHR = <GET,PUT,UPI>,-
0B2C 510 FNM = <UETINIDEV.DAT>
0B7C 511
0B7C 512 INI_RAB: ; Allocate RAB for UETINIDEV
0B7C 513 $RAB-
0B7C 514 FAB = INI_FAB,-
0B7C 515 RBF = BUFFER,-
0B7C 516 UBF = BUFFER,-
0B7C 517 USZ = REC_SIZE
0BC0 518
0BC0 519 DDB_RFA: ; RFA storage for INI_RAB
00000BC6 0BC0 520 .BLKB 6
0BC6 521
0BC6 522 .ALIGN LONG
0BC8 523 SUP_FAB: ; Allocate FAB for UETSUPDEV
0BC8 524 $FAB-
0BC8 525 FAC = GET,-
0BC8 526 SHR = <UPI,GET>,-
0BC8 527 RAT = CR,-
0BC8 528 FOP = UFO,-
0BC8 529 FNM = <UETSUPDEV.DAT>
0C18 530
0C18 531 ;
0C18 532 ; Dummy FAB and RAB to copy to the UETP unit blocks
0C18 533 ; The following FAB and RAB must be contiguous and in this order!
0C18 534 ;
0C18 535
0C18 536 DUMMY_FAB:
0C18 537 $FAB
0C68 538
0C68 539 DUMMY_RAB:
0C68 540 $RAB RSZ = WRITE_SIZE,-
0C68 541 USZ = READ_SIZE
```



Address	Hex	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	
---------	-----	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--



```
00000000'GF 04 FB 00D9 600
0002'CF 08 AB 00E0 601
00E5 602
00F0 603
02 E1 00F0 604
66 0AD8'CF 00F2 605
00F6 606
00F6 607
00F6 608
00F6 609
45 02CA'CF E9 0112 610
0117 611
0117 612
0128 613
0128 614
0128 615
0220'CF DF 0149 616
01 DD 014D 617
0074832B 8F DD 014F 618
00000000'GF 03 FB C155 619
015C 620 20$:
```

```
CALLS #4,G^LIB$SIGNAL ; Print the startup message
BISW2 #BEGIN MSGM,FLAG ; Set flag so we don't print it again
$SETPRN,S PRCNAM = PROCESS_NAME ; Set the process name to UETDMPF00_x
BBC S^#DEV$V TRM,- ; BR if SYSS$INPUT is NOT a terminal
SYSIN FAB+FAB$DEV,20$
$GETDVI,S DEVNAM = SYSS$INPUT,- ; Get the name of...
EFN = #SS SYNCH EFN,- ; ...device which may abort test
ITMLST = INPUT ITMLST,-
IOSB = QUAD_STATUS
BLBC QUAD STATUS,20$ ; Avoid CTRL/C handler if any error
$ASSIGN,S DEVNAM = BUFFER_PTR,- ; Set up for CTRL/C AST handler
CHAN = TTCHAN
$QIOW,S CHAN = TTCHAN,- ; Enable CTRL/C AST's...
FUNC = #IOS$ SETMODE!IOSM_CTRLCAST,-
P1 = CCASTHAND
PUSHAL PROCESS_NAME ; ...and tell the user...
PUSHL #1
PUSHL #UETP$ ABORTC!STSS$K_SUCCESS ; ...how to abort gracefully...
CALLS #3,G^LIB$SIGNAL ; ...
```



```
015C 622 :  
015C 623 : From UETINIDEV.DAT and UETSUPDEV.DAT, get information which gives controller  
015C 624 : and unit configuration and lets us know if the setup to run this test was  
015C 625 : done correctly.  
015C 626 :  
015C 627 $OPEN FAB = INI_FAB,- ; Open file 'UETINIDEV.DAT'  
015C 628 ERR = RMS_ERROR  
016B 629 $CONNECT RAB = INI_RAB,- ; Connect the RAB and FAB  
016B 630 ERR = RMS_ERROR  
017A 631 $MGBLSC_S INADR = INADDRESS,- ; Connect to UETSUPDEV global section  
017A 632 RETADR = OUTADDRESS,-  
017A 633 GSDNAM = SUPDEV_GBLSEC,-  
017A 634 FLAGS = #SECSM_EXPREG  
00000978 8F 50 D1 0199 635 CMPL R0,#SS$_NOSUCHSEC ; Was the section already there?  
37 12 01A0 636 ENEQ 30$ ; BR if it was...  
01A2 637 $OPEN FAB = SUP_FAB,- ; ...else open 'UETSUPDEV.DAT'  
01A2 638 ERR = RMS_ERROR  
01B1 639 $CRMPSC_S CHAN = SUP_FAB+FAB$_STV,- ; Create the global section  
01B1 640 INADR = INADDRESS,-  
01B1 641 RETADR = OUTADDRESS,-  
01B1 642 GSDNAM = SUPDEV_GBLSEC,-  
01B1 643 FLAGS = #SECSM_EXPREG!SECSM_GBL  
56 02DE'CF 02DA'CF C3 01D9 644 30$: SUBL3 OUTADDRESS,OUTADDRESS+4,R6 ; Compute global section length  
01E1 645  
01E1 646 FIND_IT:  
01E1 647 $GET RAB = INI_RAB,- ; Get the first record  
01F0 648 ERR = RMS_ERROR  
01F4 649 PUSHAL CONT_DESC ; Make sure...  
01F8 650 PUSHAL CONT_DESC ; ...that the controller name...  
00000000'GF 02 FB 01F8 651 CALLS #2,G*STR$UPCASE ; ...is all uppercase letters  
0014'CF 44 8F 91 01FF 652 CMPB #^A/D/,BUFFER ; Is this a DDB?  
27 13 0205 653 BEQL 10$ ; Go on if not  
0014'CF 45 8F 91 0207 654 CMPB #^A/E/,BUFFER ; Is this the end of the file?  
D2 12 020D 655 BNEQ FIND_IT ; Continue on if not  
0218'CF DF 020F 657 PUSHAL DEV$SC ; Push device not supported message  
0220'CF DF 0213 658 PUSHAL PROCESS_NAME ; Parameters on the stack  
02 0217 659 PUSHL #2  
00748333 8F DD 0219 660 PUSHL #UETP$_DENOSU  
02 F0 021F 661 INSV #ST$$_ERROR,- ; Set the severity code...  
00 0221 662 #ST$$_SEVERITY,-  
6E 03 0222 663 #ST$$_SEVERITY,(SP)  
02C6'CF 6E D0 0224 664 MOVL (SP),STATUS ; ...and save it as the exit status  
04 DD 0229 665 PUSHL #4  
0953 31 022B 666 BRW ERROR_EXIT ; Exit in error  
022E 667 10$: CMPC DEVNAM_LEN,BUFFER+6,DEV_NAME ; Is this the right controller?  
022E 668 BNEQ FIND_IT ; BR if not  
0238 669 MOV C3 #6,INI_RAB+RAB$_RFA,DDB_RFA ; Save the Record File Address  
08C0'CF 0B8C'CF 06 28 023A 670 CMPB #^A/T/,BUFFER+4 ; Can we test this controller?  
0018'CF 54 8F 91 0242 671 BEQL FOUND_IT ; BR if we can...  
2F 13 0248 672 $FAO_S CTRSTR = DEAD_CTRLNAME,- ; ...and yell at user if we can't  
024A 673 OUTLEN = BUFFER_PTR,-  
024A 674 OUTBUF = FAO_BUF,-  
024A 675 P1 = #DEV$SC  
024A 676 MOVL #SS$_BADPARAM,STATUS ; Set return status  
02C6'CF 14 D0 0263 677 PUSHAL BUFFER_PTR ; ...  
000C'CF DF 0268 678
```



```
00741132 01 DD 026C 679      PUSHL #1
00741132 8F DD 026E 680      PUSHL #UETPS_TEXT!STSSK_ERROR
00741132 03 DD 0274 681      PUSHL #3
0908      31 0276 682      BRW ERROR_EXIT
0279 683
0279 684 FOUND_IT:
0279 685 $GET RAB = INI_RAB,-
0279 686 ERR = RMS_ERROR ; Get a record
01F5'CF DF 0288 687 PUSHAL CONT_DESC ; Make sure...
01F5'CF DF 028C 688 PUSHAL CONT_DESC ; ...that this line...
00000000'GF 02 FB 0290 689 CALLS #2,G*STR$UPCASE ; ...is all uppercase letters
0014'CF 55 8F 91 0297 690 CMPB #^A/U/,BUFFER ; Is this a UCB?
0014'CF 44 8F 13 029D 691 BEQL 30$ ; BR if it is
0014'CF 44 8F 91 029F 692 CMPB #^A/D/,BUFFER ; Is this a DDB?
0014'CF 45 8F 13 02A5 693 BEQL 20$ ; BR if yes
0014'CF 45 8F 91 02A7 694 CMPB #^A/E/,BUFFER ; Is this the end?
0014'CF 11 13 02AD 695 BEQL 20$ ; BR if yes
0151'CF DF 02AF 696 10$: PUSHAL ILLEGAL_REC ; Then this is an error in the record
00741132 01 DD 02B3 698 PUSHL #1 ; Push the error message
00741132 8F DD 02B5 699 PUSHL #UETPS_TEXT!STSSK_ERROR ; Push the signal name
00741132 03 DD 02BB 700 PUSHL #3 ; Push the temp arg count
08C1 31 02BD 701 BRW ERROR_EXIT ; Finish for good
0123 31 02C0 702 20$: BRW ALL_SET ; Found DDB or END
0018'CF 54 8F 91 02C3 703 30$: CMPB #^A/T/,BUFFER+4 ; Is the unit testable?
0018'CF AE 12 02C9 706 BNEQ FOUND_IT ; BR if not
0018'CF 01 DD 02CB 707 PUSHL #1 ; Flag to ignore blanks when converting
0018'CF 02 DD 02CD 708 PUSHL #2 ; Set byte size of results
0018'CF 02E2'CF DF 02CF 709 PUSHAL UNIT_NUMBER ; Set address to receive word
0018'CF 01ED'CF DF 02D3 710 PUSHAL UNIT_DESC ; Push string address
00000000'GF 04 FB 02D7 711 CALLS #4,G*OTSS$CVT_TI_L ; Convert ASCII unit # to decimal
0018'CF CE 50 E9 02DE 712 BLBC R0,10$ ; Don't allow bogus unit to pass
0018'CF 05 20 3B 02E1 713 SKPC #^A/ /,MAX_UNIT_DESIG,- ; Find out where unit number really is
0018'CF 001A'CF 02E4 714 BUFFER+6
0018'CF 50 D7 02E7 715 DECL R0 ; Units must all be at least one digit
0018'CF 61 50 3B 02E9 716 SKPC #^A/O/,R0,(R1) ; Skip leading zeroes on the unit
0018'CF 02E4'CF 50 D6 02ED 717 INCL R0 ; Compensate for DECL above
0018'CF 52 02E4'CF 3C 02F7 718 ADDW3 R0,DEVNAM_LEN,DEVDSK ; Calculate device unit string length
0018'CF 0237'C2 61 50 28 02FC 719 MOVZWL DEVNAM_LEN,R2 ; Offset to unit number in DEVDSK
0018'CF 0302 721 MOVW3 R0,(R1),DEV_NAME(R2) ; Append unit number to device
0018'CF 0302 722 $GETDEV_S DEVNAM = DEVDSK,- ; Get the device characteristics
0018'CF 0317 723 MOVZBL DIBBUF+DIB$B_DEVCLASS,R7 ; Save the device class
0018'CF 031C 724 MOVZBL DIBBUF+DIB$B_DEVTYPE,R8 ; Save the device type
0018'CF 0321 725 $FAO_S CTRSTR = CS1,-
0018'CF 0321 726 OUTBUF = FAO_BUF,-
0018'CF 0321 727 P1 = R7,-
0018'CF 0321 728 P2 = R8 ; Make it into a string
0018'CF 0336 729 MATCHC #6,BUFFER,R6,@OUTADDRESS ; Find the device class and type
0018'CF 1E 13 033F 730 BEQL 40$ ; BR if it was found
0018'CF 0341 731 $FAO_S CTRSTR = CS3,- ; Try for full class support
0018'CF 0341 732 OUTBUF = FAO_BUF,-
0018'CF 0341 733 P1 = R7
0018'CF 0354 734 MATCHC #6,BUFFER,R6,@OUTADDRESS ; Find the device class only
0018'CF 0D 12 035D 735 BNEQ 50$ ; BR if not found
```



55	000F'CF	9A	035F	736	40\$:			
0017'CF	63	55	29	035F	737	MOVZBL	TEST_NAME,R5	; Get the test name length
	1F	13	0364	738	CMPC3	R5,(R3),TEST_NAME+8		; Are we the right test?
			036A	739	BEQL	60\$		; BR if yes
			036C	740				
	0218'CF	DF	036C	741	PUSHAL	DEVDS		; Push device not supported message
	0220'CF	DF	0370	742	PUSHAL	PROCESS_NAME		; Parameters on the stack
		DD	0374	743	PUSHL	#2		; Push the argument count
00748333	8F	DD	0376	744	PUSHL	#UETP\$_DENOSU		
	02	FO	037C	745	INSV	#STSSK_ERROR,-		
	00		037E	746		#STSSV_SEVERITY,-		
6E	03		037F	747		#STSS\$SEVERITY,(SP)		; Set the severity code...
02C6'CF	6E	DO	0381	748	MOVL	(SP),STATUS		; ...and save it as the exit status
	04	DD	0386	749	PUSHL	#4		; Push the partial arg count...
07F6	31	0388	750	BRW	ERROR_EXIT			; ...and split this scene



```

038B 752 :+ The following code dynamically allocates enough memory for a unit block,  

038B 753 : a device dependent parameter area and I/O buffers. The unit block is inserted  

038B 754 : into the queue header UNIT_LIST. It then initializes the unit block.  

038B 755 : A comment indicates where the device dependent parameters should be  

038B 756 : initialized. The unit block format is as follows:  

038B 757 :  

038B 758 :  

038B 759 :      UETUNT$L_FLINK +-----+  

038B 760 :      |               |  

038B 761 :      +-----+  

038B 762 :      UETUNT$L_BLINK |  

038B 763 :      |               |  

038B 764 :      +-----+  

038B 765 :      UETUNT$B_TYPE  |  

038B 766 :      |               | contains DEVDEP_SIZE + UETUNT$C_INDSIZ  

038B 767 :      +-----+  

038B 768 :      UETUNT$B_FLAGS |  

038B 769 :      |               |  

038B 770 :      +-----+  

038B 771 :      UETUNT$W_CHAN  |  

038B 772 :      |               |  

038B 773 :      +-----+  

038B 774 :      UETUNT$W_FUNC  |  

038B 775 :      |               |  

038B 776 :      +-----+  

038B 777 :      UETUNT$L_ITER  |  

038B 778 :      |               |  

038B 779 :      +-----+  

038B 780 :      UETUNT$T_FILSPC|  

038B 781 :      |               |  

038B 782 :      +-----+  

038B 783 :      NAM$C_MAXRSS bytes  

038B 784 :      |               |  

038B 785 :      +-----+  

038B 786 :      UETUNT$K_FAB   |  

038B 787 :      |               |  

038B 788 :      +-----+  

038B 789 :      FAB$C_BLN bytes  

038B 790 :      |               |  

038B 791 :      +-----+  

038B 792 :      UETUNT$K_RAB   |  

038B 793 :      |               |  

038B 794 :      +-----+  

038B 795 :      RAB$C_BLN bytes  

038B 796 :      |               |  

038B 797 :      +-----+  

038B 798 :      UETUNT$K_DEVDEP|  

038B 799 :      |               |  

038B 800 :      +-----+  

038B 801 :      user defined    |  

038B 802 :      |               |  

038B 803 :      +-----+  

038B 804 :      READ/WRITE buffers  

038B 805 :      |               |  

038B 806 :      +-----+  

038B 807 :      user defined    |  

038B 808 :      |               |  

038B 809 :      +-----+  

038B 810 :  


```



```
038B 808 60$:
038B 809
038B 810
0A88'CF 0A90'DF 5D 039C 811
56 0A90'CF D0 03A3 812
08 A6 01 90 03A8 813
01A4 8F B0 03AC 814
09 A6 03B0 815
14 A6 0218'CF 90 03B2 816
021C'DF 0218'CF 28 03B8 817
15 A6 03BF 818
0094 8F 28 03C1 819
0110 C6 0C18'CF 03C5 820
57 0110 C6 DE 03CB 821
58 0160 C6 DE 03D0 822
3C A8 57 D0 03D5 823
14 A6 90 03D9 824
34 A7 03DC 825
15 A6 DE 03DE 826
2C A7 03E1 827
03E3 828
03E3 829
03E3 830
FE93 31 03E3 831

$EXPREG_S PAGCNT = #PAGES,- ; Get a new node of demand zero memory
RETADR = NEW_NODE
INSQTI @NEW_NODE,UNIT_LIST ; Put the new node in the unit list
MOVL NEW_NODE,R6 ; Save a copy of its address
MOVB #1,DETUNT$B TYPE(R6) ; Set the structure type
MOVW #UETUNT$C INDSIZ+DEVDEP_SIZE,- ; Set the structure size
UETUNT$W SIZE(R6)
MOVB DEVDSK,UETUNT$T FILSPC(R6) ; Set the device name size
MOVC3 DEVDSK,@DEVDSK+4,-
UETUNT$T FILSPC+1(R6) ; Save the device name
MOVC3 #FAB$C BFN+RAB$C BLN,-
DUMMY FAB,UETUNT$C FAB(R6) ; Save a FAB and a RAB away
MOVAL UETUNT$K_FAB(R6),R7 ; Save the FAB address
MOVAL UETUNT$K_RAB(R6),R8 ; Save the RAB address
MOVL R7,RAB$K_FAB(R8) ; Set the FAB address in the RAB
MOVB UETUNT$T FILSPC(R6),-
FAB$B FNS(R7) ; Set the FNS field in the FAB
MOVAL UETUNT$T FILSPC+1(R6),-
FAB$K_FNA(R7) ; Set the FNA field in the FAB
; Set the device dependent parameters in here
BRW FOUND_IT ; Do the next UCB
```



```
03E6 833 :  
03E6 834 : Arrive here when we have the device configuration. In normal or loop forever  
03E6 835 : mode, set a timer far enough in the future such that we can do a reasonable  
03E6 836 : set of tests before the timer expires, but if our device gets hung, the  
03E6 837 : program won't waste too much time before noticing. Let one-shot mode be a  
03E6 838 : special case.  
03E6 839 :  
03E6 840 ALL_SET:  
0A88'CF D5 03E6 841 TSTL UNIT_LIST ; Anything to test?  
16 12 03EA 842 BNEQ 10$ ; BR if yes  
012B'CF DF 03EC 843 PUSHAL NOUNIT_SELECTED ; Else set up the error message...  
01 DD 03F0 844 PUSHL #1 ; ...argument count...  
00741132 8F DD 03F2 845 PUSHL #UETP$_TEXT!STSSK_ERROR ; ...signal name...  
03 DD 03F8 846 PUSHL #3 ; ...and parameter count  
02C6'CF 14 D0 03FA 847 MOVL #SS$_BADPARAM,STATUS ; Set return status  
077F 31 03FF 848 BRW ERROR_EXIT ; ...and give up, complaining  
0402 849 10$:  
0002'CF 04 AB 0402 850 BISW2 #SAFE_TO_UPDM,FLAG ; OK safe to update UETINIDEV.DAT now
```



```
0407 852 .SBTTL Test the DMP/DMF
0407 853
0407 854 START_TEST:
0407 855
0407 856 $ASSIGN_S - ; Assign channel to the device
0407 857 DEVNAM = DEVDSC,-
0407 858 CHAN = XD_CHAN
0418 859
0418 860
0418 861 BLBS R0,10$ ; BR if no failure
0418 862 MOVL R0,STATUS ; Save the failure status
0420 863 PUSHL STATUS ; Push the error code...
0424 864 PUSHL STATUS
0428 865 PUSHAL DEVDSC ; ...and the device designation...
042C 866 PUSHAL TEST_NAME ; ...and the test name...
0430 867 PUSHL #3 ; ...and the arg count...
0432 868 PUSHL #UETP$_DEUNUS!ST$K_ERROR ; ...and the signal name...
0438 869 PUSHL #6 ; ...and the total argument count...
043A 870 BRW ERROR_EXIT ; ...and bail out completely
043D 871 10$:
043D 872 ;
043D 873 ; Set up P1 device char buffer, P2 buffer is set up in Read/write section
043D 874
043D 875 RESTART:
043D 876 MOVAL P1BUF+2,R3 ; Address of device char for p1
0442 877 MOVW #MAX_MSG_LEN,(R3)+ ; Maximum message length
0447 878 MOVB #XMSM_CHR_LOOPB,(R3) ; Set loop back mode in char
044A 879
044A 880 $SETIMR_S - ; Set up half minute timer
044A 881 DAYTIM = HALFMIN,- ; to prevent hung
044A 882 ASTADR = TIME_ERR_OUT,-
044A 883 REQIDT = #START_TO_MSG
0461 884 START_CONT:
0461 885 $QIOW_S - ; Start the controller
0461 886 CHAN = XD_CHAN,-
0461 887 FUNC = #IOS SETMODE!IOSM_CTRL!IOSM_STARTUP,-
0461 888 IOSB = XD_IOSB,-
0461 889 ASTADR = CHK_QIO_AST,-
0461 890 ASTPRM = #START_CONT_PRM,-
0461 891 P1 = P1BUF,-
0461 892 P2 = #P2BUF_DESC,-
0461 893 P3 = #RECVPOOL_SIZ
0492 894
0492 895 START_TRI:
0492 896 $QIOW_S - ; Start the tributary
0492 897 CHAN = XD_CHAN,-
0492 898 FUNC = #IOS SETMODE!IOSM_STARTUP,-
0492 899 IOSB = XD_IOSB,-
0492 900 ASTADR = CHK_QIO_AST,-
0492 901 ASTPRM = #START_TRIB_PRM,-
0492 902 P1 = TR_P1BUF,-
0492 903 P2 = #TR_P2BUF_DESC,-
0492 904 P3 = #RECVPOOL_SIZ ; Common receive pool = 4 buffer
04C3 905
0002'CF 20 A8 04C3 906 BISW2 #FLAG_SHUTDNM,FLAG ; Set flag to say shut down the
04C8 907 ; device if errors occur
04C8 908
```



```
0014'CF 20 8A 04C8 909 $SCANTIM_S REQIDT = #START_TO_MSG ; Cancel hung timer
0014'CF 4F 8F 91 04D7 910
0002'CF 0D 12 04D7 911 $STRNLOG_S LOGNAM = MODE,- ; Get the run mode
0002'CF 02 A8 04D7 912 RSLLEN = BUFFER_PTR,-
0002'CF 10 A8 04D7 913 RSLBUF = FAO_BUF
0013 31 0507 914
050A 915 BICB2 #LC BITM,BUFFER ; Convert to upper case
050A 916 CMPB #^A70/,BUFFER ; Is this a one shot?
051D 917 BNEQ 10$ ; BR if not
051D 918 BISW2 #TEST_OVERM,FLAG ; End after one iteration
051D 919 BISW2 #MODE_IS_ONEM,FLAG ; Set mode is 'ONE' flag
051D 920 BRW LOOPBACK_TEST ; Skip the 3 min timer, mode is 'one'
10$: 921 ; Not one shot
$SETIMR_S DAYTIM = THREEMIN,- ; Set 3 minutes timer for xmit/rcv
ASTADR = TIME_SUC_OUT ; The test will do xmit/rcv for about
; 3 minutes
; Loopback test transmit and receive random data with different message length
; LOOPBACK TEST:
52 AA 8F 9A 051D 929 MOVZBL #^XAA,R2 ; Random number 1
53 2E 9A 0521 930 MOVZBL #^X2E,R3 ; Random number 2
57 00000200 8F D0 0524 931 MOVL #MAX_MSG_LEN,R7 ; Maximum message length
052B 932 SET_XMIT_BUF: ; Set up transmit buffer
052B 933 MOVAL XMIT_BUF,R6 ; Transmit buffer address
54 57 D0 0530 934 MOVL R7,R4 ; Message length in bytes
10$: 935
52 53 C0 0533 936 ADDL2 R3,R2 ; Random number as data
86 52 90 0536 937 MOVB R2,(R6)+ ; Fill in the transmit buffer
F7 54 F5 0539 938 SOBGTR R4,10$ ; Branch if more bytes to be filled
053C 939
053C 940 $SETIMR_S - ; Set half minute timer to prevent hung
053C 941 DAYTIM = HALFMIN,-
053C 942 ASTADR = TIME_ERR_OUT,-
053C 943 REQIDT = #RW_TO_MSG
0553 944
58 10 D0 0553 945 MOVL #LIMIT,R8 ; Loop 16 times for each msg length
0556 946 XMIT:
0556 947 $QIO_S - ; Transmit data message
0556 948 EFN = #XMIT EFN,- ; Event flag
0556 949 CHAN = XD CHAN,- ; Channel
0556 950 FUNC = #IOS WRITEVBLK,- ; Transmit
0556 951 IOSB = XD IOSB,- ; IOSB
0556 952 ASTADR = CHK_QIO_AST,- ; Completion ast routine
0556 953 ASTPRM = #WRITE_PRM,- ; Ast parameter
0556 954 P1 = XMIT_BUF,- ; Addr of transmit buffer
0556 955 P2 = R7 ; message length in bytes
0581 956
0581 957 RECV:
0581 958 $QIO_S - ; Read data message
0581 959 EFN = #RECV EFN,- ; Event flag
0581 960 CHAN = XD CHAN,- ; Channel
0581 961 FUNC = #IOS READVBLK,- ; Receive message
0581 962 IOSB = RCV IOSB,- ; IOSB
0581 963 ASTADR = RECV_AST,- ; Completion ast to check data received
0581 964 ASTPRM = R7,- ; Ast parameter = message length
0581 965 P1 = RECV_BUF,- ; Receive buffer
```



```
0581 966 P2 = R7 ; Message length in bytes
05A8 967
02E6'CF D6 05A8 968 INCL ITERATION ; Increment iteration count
05AC 969
05AC 970 $WAITFR_S EFN = #XMIT_EFN ; Wait until transmit done
05B5 971
9E 58 F5 05B5 972 SOBGTR R8,XMIT ; Loop for 16 times
05B8 973
05B8 974 $CANTIM_S - ; Cancel hung timer
05B8 975 -REQIDT = #RW_TO_MSG
05C7 976
09 0002'CF 01 E0 05C7 977 BBS #TEST_OVERV,FLAG,SENSE_TEST ; Is the test over?
03 57 F5 05CD 978 SOBGTR R7,10$ ; Decrement message length by one and
05D0 979 ; try again
FF4A 31 05D0 980 BRW LOOPBACK_TEST ; Re-try from beginning
FF55 31 05D3 981 10$: BRW SET_XMIT_BUF ; Set new data in transmit buffer
05D6 982
SENSE_TEST:
05D6 983 $QIOW_S - ; Read device (trib.) characteristic
05D6 984
05D6 985 CHAN = XD_CHAN,-
05D6 986 FUNC = #10$ SENSEMODE,-
05D6 987 IOSB = XD_IOSB,-
05D6 988 P1 = SENSE_P1BUF,-
05D6 989 P2 = #SENSE_P2DESC
05FD 990
$FAO_S CTRSTR = SENSE_PRM,-
05FD 991 OUTLEN = ALT_BUFFER_PTR,-
05FD 992 OUTBUF = ALT_FAO_BUF,-
05FD 993 P1 = #DEVDSCT
05FD 994
0116'CF DF 0616 995 PUSHAL ALT_BUFFER_PTR
08A5'CF 01 FB 061A 996 CALLS #1,CHECK_IOSB ; Check status
061F 997
54 0672'CF 3C 061F 998 MOVZWL XD_IOSB+2,R4 ; Number of bytes returned for p2 buff
55 03B2'CF DE 0624 999 MOVAL TR_P2BUF,R5 ; Address of P2 buff
57 06 DO 0629 1000 MOVL #TR_P2BUF_LEN,R7 ; P2 length
062C 1001 10$:
56 03C8'CF DE 062C 1002 MOVAL SENSE_P2BUF,R6 ; Address of P2 buff returned
66 54 65 06 39 0631 1003 MATCHC #6,(R5),R4,(R6) ; Check the parameters returned
OC 12 0636 1004 BNEQ 30$ ; Br if not match
55 06 A5 DE 0638 1005 MOVAL 6(R5),R5 ; Next parameter
57 06 C2 063C 1006 SUBL2 #6,R7 ; Index
EB 12 063F 1007 BNEQ 10$ ; Br if more parameters to check
001F 31 0641 1008 BRW ERROR_TEST ; Otherwise go to test error case
0644 1009 30$:
0644 1010 $FAO_S CTRSTR = SENSE_ERRMSG,-
0644 1011 OUTLEN = BUFFER_PTR,-
0644 1012 OUTBUF = FAO_BUF,-
0644 1013 P1 = (R5),-
0644 1014 P2 = 2(R5)
000C'CF DF 065C 1015 PUSHAL BUFFER_PTR ; Error message
0235 31 0660 1016 BRW FAIL_OUT ; Failure exit
0663 1017
0663 1018 ERROR_TEST:
0663 1019 $SETSFM_S ENBFLG = #0 ; Turn off system service mode
066C 1020
066C 1021 ;
066C 1022 ; Read data with IOSM_NOW specified but no data available
```



```
066C 1023 :  
066C 1024 :  
066C 1025 :  
066C 1026 :  
066C 1027 :  
066C 1028 :  
066C 1029 :  
0693 1030 :  
57 00000870 8F DO 0693 1031 :  
58 0670 CF 3C 069A 1032 :  
57 58 B1 069F 1033 :  
3B 12 06A2 1034 :  
06A4 1035 :  
06A4 1036 :  
06A4 1037 :  
06A4 1038 :  
06A4 1039 :  
06A4 1040 :  
06A4 1041 :  
06A4 1042 :  
06A4 1043 :  
06A4 1044 :  
06A4 1045 :  
06CB 1046 :  
57 00000601 8F DO 06CB 1047 :  
58 0670 CF 3C 06D2 1048 :  
57 58 B1 06D7 1049 :  
03 12 06DA 1050 :  
0099 31 06DC 1051 :  
06DF 1052 :  
06DF 1053 :  
06DF 1054 :  
0014 CF 04AF CF 28 06E3 1055 :  
04B7 CF A3 06E9 1056 :  
59 00FA 8F 06ED 1057 :  
000C CF 59 3C 06F1 1058 :  
0010 CF 53 DO 06F6 1059 :  
06FB 1060 :  
06FB 1061 :  
06FB 1062 :  
0010 CF 000C CF C0 0710 1063 :  
59 000C CF A2 0717 1064 :  
04E7 CF 28 071C 1065 :  
04EF CF 0720 1066 :  
0010 DF 0723 1067 :  
0010 CF 53 DO 0726 1068 :  
59 04E7 CF A2 072B 1069 :  
000C CF 59 3C 0730 1070 :  
0735 1071 :  
0735 1072 :  
0735 1073 :  
59 59 000C CF A2 074A 1074 :  
59 00FA 8F 59 A3 074F 1075 :  
59 59 3C 0755 1076 :  
0014 C9 2E22 8F B0 0758 1077 :  
000C CF 59 02 A1 075F 1078 :  
0010 CF 0014 CF DE 0765 1079 :  
  
$QIOW_S - ; Read data message  
CHAN = XD_CHAN,-  
FUNC = #IOS$ READVBLK!IOSM_NOW,-  
IOSB = XD_IOSB,-  
P1 = RECV_BUF,-  
P2 = #128-  
  
MOVL #SS$ ENDOFFILE,R7  
MOVZWL XD_IOSB,R8  
CMPW R8,R7 ; Correct error code?  
BNEQ ERRST_ERR ; Br if not  
  
; Buffer not enough to hold all information from IOS_SENSEMODE  
  
$QIOW_S - ; Read device (trib. ) charracteristic  
CHAN = XD_CHAN,-  
FUNC = #IOS$ SENSEMODE,-  
IOSB = XD_IOSB,-  
P1 = SENSE_P1BUF,-  
P2 = #ERRST_P2DESC  
  
MOVL #SS$ BUFFEROVF,R7  
MOVZWL XD_IOSB,R8  
CMPW R8,R7 ; Error code = buffer overflow?  
BNEQ ERRST_ERR ; Error if not  
BRW READ_ERRCOUNT ; Br to read and clear error count  
  
ERRST_ERR:  
MOVC3 COMP_STATUS_MSG,- ; We need an error message...  
COMP_STATUS_MSG+8,BUFFER  
SUBW3 COMP_STATUS_MSG,- ; ...to compare...  
#TEXT_BUFFER,R9  
MOVZWL R9,BUFFER_PTR  
MOVL R3,BUFFER_PTR+4  
$GETMSG_S MSGID = R7,- ; ...the error we expected...  
MSGLEN = BUFFER_PTR,-  
BUFADR = BUFFER_PTR  
ADDL2 BUFFER_PTR,BUFFER_PTR+4  
SUBW2 BUFFER_PTR,R9  
MOVC3 RECEIVED_MSG,-  
RECEIVED_MSG+8,-  
@BUFFER_PTR+4  
MOVL R3,BUFFER_PTR+4  
SUBW2 RECEIVED_MSG,R9  
MOVZWL R9,BUFFER_PTR  
$GETMSG_S MSGID = R8,- ; ...with the one we received  
MSGLEN = BUFFER_PTR,-  
BUFADR = BUFFER_PTR  
SUBW2 BUFFER_PTR,R9  
SUBW3 R9,#TEXT_BUFFER,R9  
MOVZWL R9,R9  
MOVW #^A/'./,BUFFER(R9)  
ADDW3 #2,R9,BUFFER_PTR  
MOVAL BUFFER,BUFFER_PTR+4
```



```
000C'CF DF 076C 1080 PUSHAL BUFFER_PTR ; Error message
02C6'CF 58 DO 0770 1081 MOVL R8,STATUS ; Save our actual error as exit status
0120 31 0775 1082 BRW FAIL_OUT ; Failure exit
0778 1083
0778 1084 READ_ERRCOUNT:
0778 1085 $SETSFM_S ENBFLG = #1 ; Turn on system service mode
0781 1086
0781 1087 $QIOW_S - ; Read and clear the error counters
0781 1088 CHAN = XD_CHAN,-
0781 1089 FUNC = #IOS_SENSEMODE!IOSM_RD_COUNT!IOSM_CLR_COUNT,-
0781 1090 IOSB = XD_IOSB,-
0781 1091 P2 = #ERRCOUNT_DESC
07A6 1092
07A6 1093 $FAO_S CTRSTR = SENSE_PRM,-
07A6 1094 OUTLEN = ALT_BUFFER_PTR,-
07A6 1095 OUTBUF = ALT_FAO_BUF,-
07A6 1096 P1 = #DEVDSCL
0116'CF DF 07BF 1097 PUSHAL ALT_BUFFER_PTR
08A5'CF 01 FB 07C3 1098 CALLS #1,CHECK_IOSB ; Check status
07C8 1099
07C8 1100 CLEAN_EXIT:
07C8 1101 BICW2 #FLAG_SHUTDNM,FLAG ; Clear the shutdown flag
07CD 1102
07CD 1103 $QIOW_S - ; Shut down the device
07CD 1104 CHAN = XD_CHAN,-
07CD 1105 FUNC = #IOS_SEMODE!IOSM_CTRL!IOSM_SHUTDOWN,-
07CD 1106 IOSB = XD_IOSB
07EE 1107
07EE 1108 $FAO_S CTRSTR = SET_PRM,-
07EE 1109 OUTLEN = ALT_BUFFER_PTR,-
07EE 1110 OUTBUF = ALT_FAO_BUF,-
07EE 1111 P1 = #DEVDSCL
07EE 1112
0116'CF DF 0807 1113 PUSHAL ALT_BUFFER_PTR
08A5'CF 01 FB 080B 1114 CALLS #1,CHECK_IOSB ; Check status
0810 1115
0810 1116 SUC_EXIT:
0810 1117 $TRNLOG_S LOGNAM = MODE,-
0810 1118 RSLLEN = BUFFER_PTR,-
0810 1119 RSLBUF = FAO_BUF ; Get the run mode
0014'CF 20 8A 0829 1120 BICB2 #LC_BITM,BUFFER ; Convert to upper case
0014'CF 4C 8F 91 082E 1121 CMPB #^A7L/,BUFFER ; Is this a loop for ever?
40 12 0834 1122 BNEQ 10$ ; BR if not
0002'CF 02 AA 0836 1123 BICW2 #TEST_OVERM,FLAG ; Reset the termination flag
02EA'CF D6 083B 1124 INCL PASS ; Bump the pass count
083F 1125 $FAO_S CTRSTR = PASS_MSG,-
083F 1126 OUTLEN = BUFFER_PTR,-
083F 1127 OUTBUF = FAO_BUF,-
083F 1128 P1 = PASS,-
083F 1129 P2 = ITERATION,-
083F 1130 P3 = #0 ; Make the end of pass message
000C'CF DF 085C 1131 PUSHAL BUFFER_PTR ; Push the string desc.
01 DD 0860 1132 PUSHL #1 ; Push arg count
00741133 8F DD 0862 1133 PUSHL #UETPS_TEXT!STSSK_INFO ; Push the signal name
00000000'GF 03 FB 0868 1134 CALLS #3,G^LIB$SIGNAL ; Print the end of pass message
02E6'CF D4 086F 1135 CLRL ITERATION ; Reset the iteration count
FBC7 31 0873 1136 BRW RESTART ; Do the next pass
```



```
56 0A88'CF 00000A88'8F C1 0876 1137 10$:
                                02 88 0876 1138 ADDL3 #UNIT_LIST,UNIT_LIST,R6 ; Set the unit block list header
                                0B A6 88 0880 1139 BISB2 #UETUNT$M TESTABLE,-
02C6'CF 10000001 8F D0 0882 1140 UETUNT$B FLAGS(R6) ; Set the testable bit
                                0884 1141 MOVL #$$$ NORMAL!STSSM_INHIB_MSG,STATUS ; Set successful exit status
                                088D 1142 $EXIT,S STATUS ; Exit with the status
                                0898 1143
                                0898 1144 FAIL_OUT: ; Failure exit
                                01 DD 0898 1145 PUSHL #1 ; Arg count
00741132 8F DD 089A 1146 PUSHL #UETPS_TEXT!STSSK_ERROR ; Signal name
                                03 DD 08A0 1147 PUSHL #3 ; Arg count
                                02DC 31 08A2 1148 BRW ERROR_EXIT ; Error exit
                                08A5 1149
```



```
08A5 1151 .SBTTL CHECKIOSB - Check IO status block
08A5 1152 :++
08A5 1153 : FUNCTIONAL DESCRIPTION:
08A5 1154 :   This routine checks the IO status block = #SS$_NORMAL
08A5 1155 :
08A5 1156 : CALLING SEQUENCE:
08A5 1157 :   CALLS #1,CHECK_IOSB
08A5 1158 :
08A5 1159 : INPUT PARAMETERS:
08A5 1160 :   Address of error message
08A5 1161 :
08A5 1162 : IMPLICIT INPUTS:
08A5 1163 :   XD_IOSB is the IOSB from some $QIO
08A5 1164 :
08A5 1165 : OUTPUT PARAMETERS:
08A5 1166 :   NONE
08A5 1167 :
08A5 1168 : IMPLICIT OUTPUTS:
08A5 1169 :   Exit with status if IOSB not right
08A5 1170 :
08A5 1171 : COMPLETION CODES:
08A5 1172 :   IO status in STATUS if error
08A5 1173 :
08A5 1174 : SIDE EFFECTS:
08A5 1175 :   Program exit if error found
08A5 1176 :
08A5 1177 :--
08A5 1178
08A5 1179 CHECK_IOSB:
01 0670'CF 0004 08A5 1180 .WORD ^M<R2>
01 01 12 08A7 1181 CMPW XD_IOSB,#SS$_NORMAL ; Is the QIO O.K.?
01 04 04 08AC 1182 BNEQ 10$ ; Br if not
08AE 1183 RET ; Return
08AF 1184 10$:
7E 0670'CF 3C 08AF 1185 MOVZWL XD_IOSB,-(SP) ; Push the error status code
02C6'CF 6E D0 08B4 1186 MOVL (SP),STATUS ; Set return status
52 0411'CF DE 08B9 1187 MOVAL DMP_IOSB_DUMP,R2 ; Assume we're testing a DMP
00' 91 08BE 1188 CMPB S^#DTS_DMP11,- ; But are we?
0253'CF 08C0 1189 DIBBUF+DIB$_DEVTYPE
05 13 08C3 1190 BEQL 20$
52 0393'CF DE 08C5 1191 MOVAL DMF_IOSB_DUMP,R2 ; Get a different string if not
08CA 1192 20$:
08CA 1193 $FAO_S CTRSTR = (R2),- ; Get the IOSB in plain text
08CA 1194 OUTLEN = BUFFER_PTR,-
08CA 1195 OUTBUF = FAO_BUF,-
08CA 1196 P1 = @XD_IOSB,-
08CA 1197 P2 = @XD_IOSB+2,-
08CA 1198 P3 = @XD_IOSB+4,-
08CA 1199 P4 = @XD_IOSB+5,-
08CA 1200 P5 = @XD_IOSB+6,-
08CA 1201 P6 = @XD_IOSB+7
000C'CF DF 08FF 1202 PUSHAL BUFFER_PTR
01 DD 0903 1203 PUSHL #1
00741132 8F DD 0905 1204 PUSHL #UETP$_TEXT!ST$K_ERROR
04 AC DD 090B 1205 PUSHL 04(AP)
01 DD 090E 1206 PUSHL #1
00741132 8F DD 0910 1207 PUSHL #UETP$_TEXT!ST$K_ERROR
```



UETDMPF00  
V04-001

VAX/VMS UETP DEVICE TEST FOR DMP 11/ <sup>D 3</sup> DMF 16-SEP-1984 01:24:05 VAX/VMS Macro V04-00  
CHECKIOSB - Check IO status block 10-SEP-1984 12:03:55 [UETP.SRC]UETDMPF00.MAR;2

Page 30  
(12)

07	DD	0916	1208	PUSHL	#7	; Argument count
0266	31	0918	1209	BRW	ERROR_EXIT	; Error exit
		091B	1210			

UET  
V04



```
091B 1212 .SBTTL Check QIO AST Routine
091B 1213 :++
091B 1214 : FUNCTIONAL DESCRIPTION:
091B 1215 : This routine will be called as a QIO completion AST routine
091B 1216 : It checks IO status block and the AST parameter
091B 1217 :
091B 1218 : CALLING SEQUENCE:
091B 1219 : Called via AST at $QIO completion
091B 1220 :
091B 1221 : INPUT PARAMETERS:
091B 1222 : NONE
091B 1223 :
091B 1224 : IMPLICIT INPUTS:
091B 1225 : DEVDSC, ALT_BUFFER_PTR, ALT_FAO_BUF and ALT_BUFFER used in forming a
091B 1226 : potential error message.
091B 1227 :
091B 1228 : OUTPUT PARAMETERS:
091B 1229 : NONE
091B 1230 :
091B 1231 : IMPLICIT OUTPUTS:
091B 1232 : Error message if error
091B 1233 :
091B 1234 : COMPLETION CODES:
091B 1235 : IO status in STATUS if error
091B 1236 :
091B 1237 : SIDE EFFECTS:
091B 1238 : Program exit if error
091B 1239 : BUFFER_PTR and BUFFER used if error
091B 1240 :
091B 1241 :--
091B 1242 CHK_QIO_AST:
OFFC 091B 1243 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
091D 1244
091D 1245 $FAO_S CTRSTR = @04(AP),- ; Form message for CHECK_IOSB
091D 1246 OUTLEN = ALT_BUFFER_PTR,-
091D 1247 OUTBUF = ALT_FAO_BUF,-
091D 1248 P1 = #DEVDSC-
FF67 0116'CF DF 0935 1249 PUSHAL ALT_BUFFER_PTR
CF 01 FB 0939 1250 CALLS #1,CHECK_IOSB ; Go check IO status block
04 093E 1251 RET
```



```
093F 1253 .SBTTL Receive data AST routine
093F 1254 :++
093F 1255 : FUNCTIONAL DESCRIPTION:
093F 1256 : This routine will be called as receive data AST routine
093F 1257 : It checks IO status and compare the data in the receive buffer
093F 1258 : against the transmit buffer
093F 1259 :
093F 1260 : CALLING SEQUENCE:
093F 1261 : Called via AST at $QIO READ
093F 1262 :
093F 1263 : INPUT PARAMETERS:
093F 1264 : AST parameter = message length
093F 1265 :
093F 1266 : IMPLICIT INPUTS:
093F 1267 : DEVDSC and various text buffers are used in forming error messages
093F 1268 :
093F 1269 : OUTPUT PARAMETERS:
093F 1270 : NONE
093F 1271 :
093F 1272 : IMPLICIT OUTPUTS:
093F 1273 : Error message if error found
093F 1274 :
093F 1275 : COMPLETION CODES:
093F 1276 : in STATUS
093F 1277 :
093F 1278 : SIDE EFFECTS:
093F 1279 : Program exit if error found
093F 1280 :
093F 1281 :--
093F 1282 RECV_AST:
093F 1283 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
0941 1284 CMPW RCV_IOSB,#SS$_NORMAL ; Is the QIO O.K.?
0946 1285 BNEQ 10$ ; Br if not
0948 1286 CMPC3 4(AP),RCV_BUF,XMIT_BUF ; Compare the data
0951 1287 BNEQ 20$
0953 1288 RET
0954 1289 10$:
0954 1290 $FAO_S CTRSTR = READ_PRM,-
0954 1291 OUTLEN = ALT_BUFFER_PTR,-
0954 1292 OUTBUF = ALT_FAO_BUF,-
0954 1293 P1 = #DEVDSC
096D 1294 MOVQ RCV_IOSB,XD_IOSB ; Set up a copy of our error status
0974 1295 PUSHAL ALT_BUFFER_PTR
0978 1296 CALLS #1,CHECK_IOSB ; Take advantage of existing routine
097D 1297 ; Note that we will not return!
097D 1298
097D 1299 20$:
097D 1300 MOVZBL (R1),-(SP) ; Save the bad data...
0980 1301 MOVZBL (R3),-(SP) ; ...the good data...
0983 1302 SUBL3 R0,4(AP),-(SP) ; ...the offset of the mismatch...
0988 1303 MOVZWL UNIT_NUMBER,-(SP) ; ...the failing unit...
098D 1304 PUSHAQ DEVDSC ; ...the device name...
0991 1305 PUSHL #5 ; ...and the count of parameters...
0993 1306 PUSHL #UETP$_DATAER!STSSK_ERROR ; ...for our error message
0999 1307 PUSHL #7
099B 1308 BRW ERROR_EXIT
```

01 0678'CF OFFC B1 0941 1284  
0680'CF 0880'CF 04 AC 29 0946 1285  
2A 12 0948 1286  
04 0951 1287  
0953 1288  
0954 1289 10\$:  
0954 1290 \$FAO\_S CTRSTR = READ\_PRM,-  
0954 1291 OUTLEN = ALT\_BUFFER\_PTR,-  
0954 1292 OUTBUF = ALT\_FAO\_BUF,-  
0954 1293 P1 = #DEVDSC  
0670'CF 0678'CF 7D 096D 1294 MOVQ RCV\_IOSB,XD\_IOSB ; Set up a copy of our error status  
0116'CF DF 0974 1295 PUSHAL ALT\_BUFFER\_PTR  
FF28 CF 01 FB 0978 1296 CALLS #1,CHECK\_IOSB ; Take advantage of existing routine  
097D 1297 ; Note that we will not return!  
097D 1298  
097D 1299 20\$:  
7E 61 9A 097D 1300 MOVZBL (R1),-(SP) ; Save the bad data...  
7E 63 9A 0980 1301 MOVZBL (R3),-(SP) ; ...the good data...  
7E 04 AC 50 C3 0983 1302 SUBL3 R0,4(AP),-(SP) ; ...the offset of the mismatch...  
02E2'CF 3C 0988 1303 MOVZWL UNIT\_NUMBER,-(SP) ; ...the failing unit...  
0218'CF 7F 098D 1304 PUSHAQ DEVDSC ; ...the device name...  
05 DD 0991 1305 PUSHL #5 ; ...and the count of parameters...  
00748012 8F DD 0993 1306 PUSHL #UETP\$\_DATAER!STSSK\_ERROR ; ...for our error message  
07 DD 0999 1307 PUSHL #7  
01E3 31 099B 1308 BRW ERROR\_EXIT



```
099E 1310 .SBTTL Half Minute Timer Expiration Routine
099E 1311 :++
099E 1312 : FUNCTIONAL DESCRIPTION:
099E 1313 : This routine will be called only if the timer which was set to prevent
099E 1314 : program hangs goes off.
099E 1315 :
099E 1316 : CALLING SEQUENCE:
099E 1317 : Called via AST at $SETIMR expiration.
099E 1318 :
099E 1319 : INPUT PARAMETERS:
099E 1320 : 04(AP) Address of a descriptor for an error message
099E 1321 :
099E 1322 : IMPLICIT INPUTS:
099E 1323 : DEVDSK and various text buffers are used to for error messages
099E 1324 :
099E 1325 : OUTPUT PARAMETERS:
099E 1326 : NONE
099E 1327 :
099E 1328 : IMPLICIT OUTPUTS:
099E 1329 : Time out error message
099E 1330 :
099E 1331 : COMPLETION CODES:
099E 1332 : NONE
099E 1333 :
099E 1334 : SIDE EFFECTS:
099E 1335 : Program exit
099E 1336 :
099E 1337 :--
099E 1338 :
OFFC 099E 1339 TIME_ERR_OUT:
099E 1340 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
09A0 1341
09A0 1342 $FAO_S CTRSTR = @04(AP),-
09A0 1343 OUTLEN = ALT_BUFFER_PTR,-
09A0 1344 OUTBUF = ALT_FAO_BUF,-
09A0 1345 P1 = #DEVDSK
0116'CF DF 09B8 1346 PUSHAL ALT_BUFFER_PTR ; Set up our error message
01 DD 09BC 1347 PUSHL #1
00741132 8F DD 09BE 1348 PUSHL #UETP$_TEXT!ST$K_ERROR
03 DD 09C4 1349 PUSHL #3 ; Push the argument count total
01B8 31 09C6 1350 BRW ERROR_EXIT ; Bail out completely
```



```

09C9 1352 .SBTTL Three Minutes Timer Expiration Routine
09C9 1353 :++
09C9 1354 : FUNCTIONAL DESCRIPTION:
09C9 1355 : This routine will be called when the device test has been run for
09C9 1356 : about three minutes. (that is, one normal run )
09C9 1357 :
09C9 1358 : CALLING SEQUENCE:
09C9 1359 : Called via AST at $SETIMR expiration.
09C9 1360 :
09C9 1361 : INPUT PARAMETERS:
09C9 1362 : NONE
09C9 1363 :
09C9 1364 : IMPLICIT INPUTS:
09C9 1365 : NONE
09C9 1366 :
09C9 1367 : OUTPUT PARAMETERS:
09C9 1368 : NONE
09C9 1369 :
09C9 1370 : IMPLICIT OUTPUTS:
09C9 1371 : NONE
09C9 1372 :
09C9 1373 : COMPLETION CODES:
09C9 1374 : NONE
09C9 1375 :
09C9 1376 : SIDE EFFECTS:
09C9 1377 : Get out the transmit /receive loop test
09C9 1378 :
09C9 1379 :--
09C9 1380 :
09C9 1381 TIME_SUC_OUT:
09C9 1382 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
09CB 1383 BISW2 #TEST_OVERM,FLAG ; set test over bit
09D0 1384 RET

```

UETC  
Symt  
IOSM  
IOSM  
IOSM  
IOS  
IOS  
IOS  
IOS  
ITEF  
LC E  
LIBS  
LIM  
LOOP  
MAX  
MAX  
MAX  
MAX  
MODE  
MODE  
MODE  
MSG  
NAME  
NEW  
NMA  
NMA  
NMA  
NMA  
NMA  
NOUN  
NO-C  
NO-F  
NRAT  
OTS  
OUTA  
P1BL  
P2BL  
P2BL  
P2BL  
PAGE  
PASS  
PASS  
PMTS  
PRM  
PROC  
PROC  
PROC  
PROI  
QUAL  
RAB  
RAB  
RAB  
RAB  
RAB  
RAB  
RAB  
RAB  
RAB  
RAB



```
09D1 1386
09D1 1387 .SBTTL System Service Exception Handler
09D1 1388 :++
09D1 1389 : FUNCTIONAL DESCRIPTION:
09D1 1390 : This routine is executed if a software or hardware exception occurs or
09D1 1391 : if a LIB$SIGNAL system service is used to output a message.
09D1 1392 :
09D1 1393 : CALLING SEQUENCE:
09D1 1394 : Entered via an exception from the system
09D1 1395 :
09D1 1396 : INPUT PARAMETERS:
09D1 1397 : ERROR_COUNT = previous cumulative error count
09D1 1398 :
09D1 1399 : AP ---->
09D1 1400 :
09D1 1401 : SIGNAL ARY PNT
09D1 1402 :
09D1 1403 : MECH ARY PNT
09D1 1404 :
09D1 1405 : 4
09D1 1406 :
09D1 1407 : ESTABLISH FP
09D1 1408 :
09D1 1409 : DEPTH
09D1 1410 :
09D1 1411 : R0
09D1 1412 :
09D1 1413 : R1
09D1 1414 :
09D1 1415 : N
09D1 1416 :
09D1 1417 : CONDITION NAME
09D1 1418 :
09D1 1419 : N-3 ADDITIONAL
09D1 1420 : LONG WORD ARGS
09D1 1421 :
09D1 1422 : PC
09D1 1423 :
09D1 1424 : PSL
09D1 1425 :
09D1 1426 : IMPLICIT INPUTS:
09D1 1427 : NONE
09D1 1428 :
09D1 1429 : OUTPUT PARAMETERS:
09D1 1430 : NONE
09D1 1431 :
09D1 1432 : IMPLICIT OUTPUTS:
09D1 1433 : NONE
09D1 1434 :
09D1 1435 : COMPLETION CODES:
09D1 1436 : SSS_NORMAL if it's a UETP condition or RMS error.
09D1 1437 : Error status from exception, otherwise.
09D1 1438 :
09D1 1439 : SIDE EFFECTS:
09D1 1440 : May branch to ERROR_EXIT.
09D1 1441 : May print a message.
09D1 1442 :--
```

2	
SIGNAL ARY PNT	
MECH ARY PNT	
4	
ESTABLISH FP	
DEPTH	Mechanism Array
R0	
R1	
N	
CONDITION NAME	
N-3 ADDITIONAL LONG WORD ARGS	Signal Array
PC	
PSL	



```
09D1 1443
09D1 1444 SSERROR:
OFFC 09D1 1445 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
09D3 1446
09D3 1447
50 01 DD 09DC 1448 $SETAST_S ENBFLG = #0 ; Disable AST delivery
09 D1 09DE 1449 PUSHL #1 ; Assume ASTs were enabled
02 13 09E1 1450 CMPL S^SSS_WASSET,R0 ; Were ASTs enabled?
6E D4 09E3 1451 BEQL 10$ ; BR if they were
09E5 1452 CLRL (SP) ; Set ASTs to remain disabled
10$: 09E5 1453
09EE 1454 $SETSFM_S ENBFLG = #0 ; Disable SS failure mode
50 01 DD 09F0 1455 PUSHL #1 ; Assume SS failure mode was enabled
09 D1 09F0 1455 CMPL S^SSS_WASSET,R0 ; Was SS failure mode enabled?
02 13 09F3 1456 BEQL 20$ ; BR if it was
6E D4 09F5 1457 CLRL (SP) ; Set SS failure mode to remain cif
20$: 09F7 1458
09F7 1459 MOVL CHF$SIGARGLIST(AP),R6 ; Get the signal array pointer
56 04 AC D0 09F7 1459 MOVQ CHF$SIG_NAME(R6),R9 ; Get NAME in R9 and ARG1 in R10
59 04 A6 7D 09FB 1460 CMPZV #STSSV_FAC_NO,- ; Is this a message from LIB$SIGNAL?
10 ED 09FF 1461 #STSSV_FAC_NO,-
0C 0A01 1462 R9,#UETPS_FACILITY
00000074 8F 59 0A02 1463
14 12 0A08 1464
66 02 C2 0A0A 1465
0A0D 1466
21 11 0A1C 1467
0A1E 1468 30$:
59 0000045C 8F D1 0A1E 1469 CMPL #SSS_SSFAIL,R9 ; RMS failures are SysSvc failures
32 12 0A25 1470 BNEQ 50$ ; BR if this can't be an RMS failure
10 ED 0A27 1471 CMPZV #STSSV_FAC_NO,- ; Is it an RMS failure?
0C 0A29 1472 #STSSV_FAC_NO,-
01 5A 0A2A 1473 R10,#RMS$FACILITY
2B 12 0A2C 1474
5A F0000000 8F CA 0A2E 1475 BNEQ 50$ ; BR if not
08 A6 04 39 0A35 1476 BICL2 #XF0000000,R10 ; Strip control bits from status code
14 0A39 1477 MATCHC #4,CHF$SIG_ARG1(R6),- ; Is it an RMS failure for which...
004D'CF 0A3A 1478 #NRAT_LENGTH,-
1A 13 0A3D 1479 NO RMS_AST_TABLE
40$: 0A3F 1480 BEQL 50$ ; ...no AST can be delivered?
01 BA 0A3F 1481 POPR #^M<R0> ; BR if so - must give error here
0A41 1482 ; Restore SS failure mode...
01 BA 0A4A 1483 $SETSFM_S ENBFLG = R0 ; ...
0A4C 1484 POPR #^M<R0> ; Restore AST enable...
50 01 D0 0A55 1485 $SETAST_S ENBFLG = R0 ; ...
04 0A58 1486 MOVL S^SSS_NORMAL,R0 ; Supply a standard status for exit
0A59 1487 RET ; Resume processing (or goto RMS_ERROR)
50$: 0A59 1488
0A5E 1489 MOVL R9,STATUS ; Save the status
59 0000045C 8F D1 0A60 1490 CLRL R8 ; Assume for now it's not SS failure
38 12 0A67 1491 CMPL #SSS_SSFAIL,R9 ; But is it a System Service failure?
0A69 1492 BNEQ 70$ ; BR if not - no special case message
0A69 1493 $GETMSG_S MSGID = R10,- ; Get SS failure code associated text
0A69 1494 MSGLEN = BUFFER_PTR,-
0A69 1495 BUFADR = FAO_BUF,-
0A69 1496 FLAGS = #14,-
02EF'CF 95 0A69 1496 OUTADR = MSG_BLOCK
16 13 0A80 1497 TSTB MSG_BLOCK+1 ; Get FAO arg count for SS failure code
000C'CF DF 0A84 1498 BEQL 60$ ; Don't use $GETMSG if no $FAO args...
0A86 1499 PUSHAL BUFFER_PTR ; ...else build up...
```



```
00741130 01 DD 0A8A 1500 PUSHL #1 ; ...a message describing...
          8F DD 0A8C 1501 PUSHL #UETP$ TEXT ; ...why the System Service failed
          00 5A F0 0A92 1502 INSV R10,#STSSV_SEVERITY,- ; Give the message...
          6E 03 0A95 1503 ; #STSS$SEVERITY,(SP) ; ...the correct severity code
          58 03 D0 0A97 1504 MOVL #3,R8 ; Count the number of args we pushed
          05 11 0A9A 1505 BRB 70$
          0A9C 1506 60$:
          5A DD 0A9C 1507 PUSHL R10 ; Save SS failure code
          58 01 D0 0A9E 1508 MOVL #1,R8 ; Count the number of args we pushed
          0AA1 1509 70$:
          57 66 04 C5 0AA1 1510 MULL3 #4,CHF$L_SIG_ARGS(R6),R7 ; Convert longwords to bytes
          5E 57 C2 0AA5 1511 SUBL2 R7,SP ; Save the current signal array...
          6E 04 A6 57 28 0AA8 1512 MOVC3 R7,CHF$L_SIG_NAME(R6),(SP) ; ...on the stack
          7E 66 58 C1 0AAD 1513 ADDL3 R8,CHF$L_SIG_ARGS(R6),-(SP) ; Push the current arg count
          00CD 31 0AB1 1514 BRW ERROR_EXIT
```



```

OAB4 1516      .SBTTL RMS Error Handler
OAB4 1517      :++
OAB4 1518      : FUNCTIONAL DESCRIPTION:
OAB4 1519      :   This routine handles error returns from RMS calls.
OAB4 1520      :
OAB4 1521      : CALLING SEQUENCE:
OAB4 1522      :   Called by RMS when a file processing error is found.
OAB4 1523      :
OAB4 1524      : INPUT PARAMETERS:
OAB4 1525      :   The FAB or RAB associated with the RMS call.
OAB4 1526      :
OAB4 1527      : IMPLICIT INPUTS:
OAB4 1528      :   NONE
OAB4 1529      :
OAB4 1530      : OUTPUT PARAMETERS:
OAB4 1531      :   NONE
OAB4 1532      :
OAB4 1533      : IMPLICIT OUTPUTS:
OAB4 1534      :   Error message
OAB4 1535      :
OAB4 1536      : COMPLETION CODES:
OAB4 1537      :   NONE
OAB4 1538      :
OAB4 1539      : SIDE EFFECTS:
OAB4 1540      :   Program may exit, depending on severity of the error.
OAB4 1541      :
OAB4 1542      :--
OAB4 1543
OAB4 1544      RMS_ERROR:
OAB4 1545      .WORD      ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OAB6 1546
56  04 AC  DO  OAB6 1547      MOVL      4(AP),R6          ; See whether we're dealing with...
66  03 91  OABA 1548      CMPB      #FAB$C_BID,FAB$B_BID(R6) ; ...a FAB or a RAB
16 12 OABD 1549      BNEQ      10$          ; BR if it's a RAB
57  01FD'CF DE  OABF 1550      MOVAL      FILE,R7          ; FAB-specific code: text string...
58  56 DO  OAC4 1551      MOVL      R6,R8          ; ...address of FAB...
OC A6 DD  OAC7 1552      PUSHL     FAB$STV(R6)        ; ...STV field for error...
08 A6 DD  OACA 1553      PUSHL     FAB$STS(R6)        ; ...STS field for error...
02C6'CF 08 A6 DO  OACD 1554      MOVL     FAB$STS(R6),STATUS ; ...and save the error code
15 11 OAD3 1555      BRB        COMMON          ; FAB and RAB share other code
OAB5 1556      10$:
57  0209'CF DE  OAD5 1557      MOVAL      RECORD,R7        ; RAB-specific code: text string...
58  3C A6 DO  OADA 1558      MOVL     RAB$FAB(R6),R8      ; ...address of associated FAB...
OC A6 DD  OADE 1559      PUSHL     RAB$STV(R6)        ; ...STV field for error...
08 A6 DD  OAE1 1560      PUSHL     RAB$STS(R6)        ; ...STS field for error...
02C6'CF 08 A6 DO  OAE4 1561      MOVL     RAB$STS(R6),STATUS ; ...and save the error code
OAEA 1562      COMMON:
5A  34 A8 9A  OAEA 1563      MOVZBL   FAB$B_FNS(R8),R10    ; Get the file name size
OAEF 1564      $FAO_S      CTRSTR = RMS_ERR_STRING,- ; Common code, prepare error message...
OAEF 1565      OUTLEN = BUFFER_PTR,-
OAEF 1566      OUTBUF = FAO_BUF,-
OAEF 1567      P1 = R7,-
OAEF 1568      P2 = R10,-
OAEF 1569      P3 = FAB$L_FNA(R8)
000C'CF DF  OB08 1570      PUSHAL    BUFFER_PTR        ; ...and arguments for ERROR_EXIT...
01 DD  OB0C 1571      PUSHL     #1
00741130 8F DD  OB0E 1572      PUSHL     #UETP$TEXT        ; ...

```



59	02C6'CF	00	EF	0B14	1573	EXTZV	#STSSV_SEVERITY,-	
	6E	03		0B16	1574		#STSSS_SEVERITY,-	
		59	88	0B17	1575		STATUS,R9	; ...get the severity code...
		05	DD	0B1B	1576	BISB2	R9,(SP)	; ...and add it into the signal name
				0B1E	1577	PUSHL	#5	; Current arg count
	005E	31		0B20	1578	BRW	ERROR_EXIT	



```

OB23 1580 .SBTTL CTRL/C Handler
OB23 1581 :++
OB23 1582 : FUNCTIONAL DESCRIPTION:
OB23 1583 : This routine handles CTRL/C AST's
OB23 1584 :
OB23 1585 : CALLING SEQUENCE:
OB23 1586 : Called via AST
OB23 1587 :
OB23 1588 : INPUT PARAMETERS:
OB23 1589 : NONE
OB23 1590 :
OB23 1591 : IMPLICIT INPUTS:
OB23 1592 : NONE
OB23 1593 :
OB23 1594 : OUTPUT PARAMETERS:
OB23 1595 : NONE
OB23 1596 :
OB23 1597 : IMPLICIT OUTPUTS:
OB23 1598 : NONE
OB23 1599 :
OB23 1600 : COMPLETION CODES:
OB23 1601 : NONE
OB23 1602 :
OB23 1603 : SIDE EFFECTS:
OB23 1604 : NONE
OB23 1605 :
OB23 1606 :--
OB23 1607 :
OB23 1608 CCASTHAND:
OB23 1609 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OB25 1610
21 0002'CF 05 E1 OB25 1611 BBC #FLAG_SHUTDNV,FLAG,10$ ; Have to shut down device?
OB2B 1612 $QIO_S - ; Shut down the device
OB2B 1613 CHAN = XD_CHAN,-
OB2B 1614 FUNC = #IOS$ SETMODE!IOSM_CTRL!IOSM_SHUTDOWN,-
OB2B 1615 IOSB = XD_IOSB
OB4C 1616 10$:
OB4C 1617 PUSHAL CNTRLMSG ; Set message pointer
OB50 1618 PUSHL #1 ; Set arg count
00741130 8F DD OB52 1619 PUSHL #UETP$_TEXT!ST$K_WARNING ; Set signal name
OB58 1620 PUSHL #0 ; Indicate an abnormal termination
0220'CF 00 DD OB5A 1621 PUSHAL PROCESS_NAME ; ...
OB5E 1622 PUSHL #2 ; ...
007410E0 8F DD OB60 1623 PUSHL #UETP$_ABENDD!ST$K_WARNING ; ...
00000000'GF 07 FB OB66 1624 CALLS #7,G^LIB$SIGNAL ; Output the message
OB6D 1625 MOVL #<ST$M_INHIB_MSG!- ; Set the exit status
OB6E 1626 $$$ CONTROLC=-
OB6E 1627 ST$K_SUCCESS+ST$K_WARNING>,-
02C6'CF 10000650 8F OB6E 1628 STATUS
OB76 1629 $EXIT_S STATUS ; Terminate program cleanly
```



```
OB81 1631 .SBTTL Error Exit
OB81 1632 :++
OB81 1633 : FUNCTIONAL DESCRIPTION:
OB81 1634 : This routine prints an error message and exits.
OB81 1635 :
OB81 1636 : CALLING SEQUENCE:
OB81 1637 : MOVx error status value,STATUS
OB81 1638 : PUSHx error specific information on the stack
OB81 1639 : PUSHL current argument count
OB81 1640 : BRW ERROR_EXIT
OB81 1641 :
OB81 1642 : INPUT PARAMETERS:
OB81 1643 : Arguments to LIB$SIGNAL, as above
OB81 1644 :
OB81 1645 : IMPLICIT INPUTS:
OB81 1646 : NONE
OB81 1647 :
OB81 1648 : OUTPUT PARAMETERS:
OB81 1649 : Message to SYS$OUTPUT and SYS$ERROR
OB81 1650 :
OB81 1651 : IMPLICIT OUTPUTS:
OB81 1652 : Program exit
OB81 1653 :
OB81 1654 : COMPLETION CODES:
OB81 1655 : Error in STATUS
OB81 1656 :
OB81 1657 : SIDE EFFECTS:
OB81 1658 : NONE
OB81 1659 :
OB81 1660 : --
OB81 1661 :
OB81 1662 : ERROR_EXIT:
OB81 1663 :
OB81 1664 : $SETAST_S ENBFLG = #0 ; ASTs can play havoc with messages
15 0002'CF 03 E0 OB8A 1665 BBS #BEGIN_MSGV,FLAG,10$ ; BR if "begin" msg already printed
OB81 1666 CLRL -(SP) ; Set the time stamp flag
OB81 1667 PUSHAL TEST_NAME ; Set the test name
OB81 1668 PUSH #2 ; Push the argument count
000F'CF 02 DD OB96 1668 PUSHL #UETP$_BEGINDD!ST$K_SUCCESS ; Set the message code
00741039 8F DD OB98 1669 CALLS #4,G^LIB$SIGNAL ; Print the startup message
00000000'GF 04 FB OB9E 1670
OB81 1671 10$:
OB81 1672 ADDL3 (SP)+,#8,ARG_COUNT ; Get total # args, pop partial count
0302'CF 08 8E C1 OBA5 1672 INCL ERROR_COUNT ; Keep running error count
OB81 1673 PUSHL #0 ; Push the time parameter
OB81 1674 PUSHAL PROCESS_NAME ; Push test name...
OB81 1675 PUSHL #^XF0002 ; ...arg count...
OB81 1676 PUSHL #UETP$_ABENDDD!ST$K_ERROR ; ...and signal name
OB81 1677 PUSHL ERROR_COUNT ; Finish off arg list...
OB81 1678 PUSHAL PROCESS_NAME ; ...
OB81 1679 PUSHL #^X10002 ; ...
OB81 1680 PUSHL #UETP$_ERBOXPROC!ST$K_ERROR ; ...for error box message
00010002 8F DD OBC9 1680 CALLS ARG_COUNT,G^LIB$SIGNAL ; Truly bitch
00748022 8F DD OBCF 1681
00000000'GF 0302'CF FB OBD5 1682
OB81 1683 OBDE 1683
OB81 1684 TSTL STATUS ; Did we exit with an error code?
OB81 1685 BNEQ 20$ ; BR if we did
007410E2 8F D0 OBE4 1685 MOVL #UETP$_ABENDDD!ST$K_ERROR,- ; Supply a generic one otherwise
OB81 1686
OB81 1687 STATUS
```



1F 0002'CF	05	E1	OBED 1688 20\$:	BBC	#FLAG_SHUTDNV,FLAG,30\$ ; Have to shut down device?
			OBED 1689	\$QIO_S -	; Shut down the device
			OBF3 1690		
			OBF3 1691		
			OBF3 1692		
			OC12 1693 30\$:		
02C6'CF	10000000 8F	C8	OC12 1694	BISL	#STSSM_INHIB_MSG,STATUS ; Don't print messages twice!
			OC1B 1695	\$EXIT_S	STATUS ; Exit in error



```
OC26 1697 .SBTTL Exit Handler
OC26 1698 :++
OC26 1699 : FUNCTIONAL DESCRIPTION:
OC26 1700 : This routine handles cleanup at exit. If the MODE logical name is
OC26 1701 : equated to 'ONE', the routine will update the test flag in the
OC26 1702 : UETINIDEV.DAT file depending on the UETUNTSM_TESTABLE flag state in the
OC26 1703 : UETUNT$B_FLAGS field of the unit block for each unit for the device
OC26 1704 : under test.
OC26 1705 :
OC26 1706 : CALLING SEQUENCE:
OC26 1707 : Invoked automatically by $EXIT System Service.
OC26 1708 :
OC26 1709 : INPUT PARAMETERS:
OC26 1710 : STATUS contains the exit status.
OC26 1711 : FLAG has synchronizing bits.
OC26 1712 : DDB_RFA contains the RFA of the DDB record for this device in UETINIDEV.
OC26 1713 :
OC26 1714 : IMPLICIT INPUTS:
OC26 1715 : UNIT_LIST points to the head of a doubly linked circular list of unit
OC26 1716 : blocks for the device under test.
OC26 1717 :
OC26 1718 : OUTPUT PARAMETERS:
OC26 1719 : NONE
OC26 1720 :
OC26 1721 : IMPLICIT OUTPUTS:
OC26 1722 : Various files are de-accessed, the process name is reset, and any
OC26 1723 : necessary synchronization with UETPDEV01 is carried out.
OC26 1724 : If the MODE logical name is equated to 'ONE', the routine will update
OC26 1725 : the test flag in the UETINIDEV.DAT file depending on the
OC26 1726 : UETUNTSM_TESTABLE flag state in the UETUNT$B_FLAGS field of the unit
OC26 1727 : block for each unit for the device under test.
OC26 1728 :
OC26 1729 : COMPLETION CODES:
OC26 1730 : NONE
OC26 1731 :
OC26 1732 : SIDE EFFECTS:
OC26 1733 : NONE
OC26 1734 :
OC26 1735 :--
OC26 1736 :
OC26 1737 EXIT_HANDLER:
OFFC OC26 1738 .WORD ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11> ; Entry mask
OC28 1739
OC28 1740 $SETSFMS ENBFLG = #0 ; Turn off System Service failure mode
OC31 1741 $SETASTS ENBFLG = #0 ; No more ASTs
OC3A 1742 $TRNLOGS LOGNAM = MODE,- ; Get the run mode
OC3A 1743 RSLLEN = BUFFER PTR,-
OC3A 1744 RSLBUF = FAO BUF
0014'CF 20 8A OC53 1745 BICB2 #LC_BITM,BUFFER ; Convert to upper case
0014'CF 4F 8F 91 OC58 1746 CMPB #^A70/,BUFFER ; Is this a one shot?
03 00 03 13 OC5E 1747 BEQL 10$ ; BR if yes...
00B8 31 OC60 1748 BRW END_UPDATE ; ...else don't update UETINIDEV.DAT
03 0002'CF 02 E0 OC63 1749 10$: BBS #SAFE TO UPDV,FLAG,20$ ; Only update if it's safe
00AF 31 OC69 1750 BRW END_UPDATE ; Else forget it
5A 0B7C'CF DE OC6C 1752 20$:
OC6C 1753 MOVAL INI_RAB,R10 ; Set the RAB address
```



```
10 AA 1E AA 02 90 0C71 1754      MOVB #RAB$C_RFA,RAB$B_RAC(R10) ; Set RFA mode
      OBC0'CF 06 28 0C75 1755      MOVBC3 #6,DDB_RFA,RAB$W_RFA(R10) ; Set RFA to DDB line
      75 50 E9 0C7C 1756      $GET RAB = (R10) ; Go back to the DDB record
      1E AA 00 90 0C85 1757      BLBC R0,UPDATE_FAILED ; If failure then forget it
5B 0A88'CF 00000A88'8F C1 0C88 1758      MOVBC #RAB$C_SEQ,RAB$B_RAC(R10) ; Set back to sequential mode
      59 D4 0C96 1760      ADDL3 #UNIT_LIST,UNIT_LIST,R11 ; Set the unit block list header
      01 E1 0C98 1761      CLRL R9 ; Init a counter
      02 0B AB E1 0C98 1762      UNIT_LOOP: BBC #UETUNT$V_TESTABLE,- ; BR if this unit is not testable
      59 D6 0C9A 1763      UETUNT$B_FLAGS(R11),10$
      0C9F 1764      INCL R9 ; Count testable units
      5B 6B C0 0C9F 1765      10$: ADDL2 (R11),R11 ; Next unit block
00000A88'8F 5B D1 0CA2 1767      CMPL R11,#UNIT_LIST ; Are we full circle in the list?
      ED 12 0CA9 1768      BNEQ UNIT_LOOP ; BR if not
      59 D5 0CAB 1769      TSTL R9 ; Any testable units?
      12 12 0CAD 1770      BNEQ 20$ ; BR if yes...
      0018'CF 4E 8F 90 0CAF 1771      MOVBC #^A/N/,BUFFER+4 ; ...else disable the DDB record...
      3C 50 E9 0CB5 1772      $UPDATE RAB = (R10) ; ...here
      0CBE 1773      BLBC R0,UPDATE_FAILED ; If error then forget it
      5B 6B C0 0CC1 1774      20$: ADDL2 (R11),R11 ; Next unit block
00000A88'8F 5B D1 0CC4 1775      CMPL R11,#UNIT_LIST ; Are we full circle in the list?
      4E 13 0CCB 1776      BEQL END_UPDATE ; BR if yes
      0CCE 1777      $GET RAB = (R10) ; Get a record
      24 50 E9 0CD6 1779      BLBC R0,UPDATE_FAILED ; If error then forget it
      0014'CF 20 8A 0CD9 1780      BICB2 #LC_BITM,BUFFER ; Convert to uppercase
      0014'CF 55 8F 91 0CDE 1781      CMPB #^A7U/,BUFFER ; Is it a UCB record?
      35 12 0CE4 1782      BNEQ END_UPDATE ; BR if not
      01 E0 0CE6 1783      BBS #UETUNT$V_TESTABLE,- ; BR if this unit is testable...
      0018'CF D6 0B AB E0 0CE8 1784      UETUNT$B_FLAGS(R11),20$
      4E 8F 90 0CEB 1785      MOVBC #^A/N/,BUFFER+4 ; ...else disable the UCB record...
      C4 50 E8 0CF1 1786      $UPDATE RAB = (R10) ; ...here
      0CFA 1787      BLBS R0,20$ ; Look at the next record if no error
      0C 50 DD 0CFD 1788      UPDATE_FAILED: PUSHL RAB$L_STV(R10) ; Do a simple message...
      01B8'CF 50 DD 0D00 1790      PUSHL R0 ; ...to tell of the failure
      01 DD 0D02 1791      PUSHAL INIDEV_UPDERR
      01 DD 0D06 1792      PUSHAL #1
      7E 50 03 0D08 1793      EXTZV #STSS$V_SEVERITY,- ; Copy the severity from RMS status...
      6E 00741130 8F C8 0D0A 1794      #STSS$S_SEVERITY,R0,-(SP)
      00000000'GF 05 FB 0D0D 1795      BISL2 #UETP$-TEXT,(SP) ; ...to our message
      0D14 1796      CALLS #5,G^LIB$SIGNAL
      00 DD 0D1B 1797      END_UPDATE: PUSHL #0 ; Set the time flag
      000F'CF DF 0D1D 1799      PUSHAL TEST_NAME ; Push the test name
      02 DD 0D21 1800      PUSHAL #2 ; Push arg count
      00 EF 0D23 1801      EXTZV #STSS$V_SEVERITY,- ; Push the proper exit severity...
      03 0D25 1802      #STSS$S_SEVERITY,-
      7E 02C6'CF 0D26 1803      STATUS,-(SP)
      6E 00741080 8F C8 0D2A 1804      BISL2 #UETP$-ENDED,(SP) ; ...and use it in our message code
      04 DD 0D31 1805      PUSHAL #4
      51 5E D0 0D33 1806      MOVL SP,R1
      0D36 1807      $PUTMSG,S MSGVEC = (R1) ; Output the message
      0D45 1808      $SETPRN,S PRCNAM = ACNT_NAME ; Reset the process name
      04 0D50 1809      RET ; That's all folks!
      0D51 1810
```



UETDMPF00  
V04-001

VAX/VMS UETP DEVICE TEST FOR DMP <sup>F 4</sup> 11/ DMF 16-SEP-1984 01:24:05 VAX/VMS Macro V04-00 Page 45  
Exit Handler 10-SEP-1984 12:03:55 [UETP.SRC]UETDMPF00.MAR;2 (22)

OD51 1811 .END UETDMPF00

UET  
V04

53

31

50

41

4E

21

2A

65  
72

6E  
63



UETDMPF00  
Symbol table

VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05 VAX/VMS Macro V04-00  
10-SEP-1984 12:03:55 [UETP.SRC]UETDMPF00.MAR;2

Page 46  
(22)

\$\$TAB	= 00000C68	R	03	END_UPDATE	00000D1B	R	05
\$\$TABEND	= 00000CAC	R	03	ERRCNT_BUF	00000470	R	03
\$\$TMP	= 00000000			ERRCNT_LEN	= 00000200		
\$\$TMP1	= 00000001			ERRCNT_DESC	00000468	R	03
\$\$TMP2	= 0000006A			ERROR_COUNT	000002C2	R	03
\$\$TMPX	= 00000016	R	04	ERROR_EXIT	00000B81	R	05
\$\$TMPX1	= 0000000D			ERROR_TEST	00000663	R	05
\$\$T1	= 00000001			ERRTEST_MSG	000005C0	R	02
\$\$T2	= 00000006			ERRTST_ERR	000006DF	R	05
ACNT_NAME	00000000	R	02	ERRTST_P2BUF	00000460	R	03
ALL_SET	000003E6	R	05	ERRTST_P2DESC	00000458	R	03
ALT_BUFFER	0000011E	R	03	ERRTST_P2LEN	= 00000008		
ALT_BUFFER_PTR	00000116	R	03	ESC	= 0000001B		
ALT_FAO_BUF	0000010E	R	03	EXIT_DESC	000002F2	R	03
ARG_COUNT	00000302	R	03	EXIT_HANDLER	00000C26	R	05
BAD_DATA	00000A80	R	03	FAB\$B_BID	= 00000000		
BEGIN_MSGM	= 00000008			FAB\$B_FNS	= 00000034		
BEGIN_MSGV	= 00000003			FAB\$C_BID	= 00000003		
BUFFER	00000014	R	03	FAB\$C_BLN	= 00000050		
BUFFER_PTR	0000000C	R	03	FAB\$C_SEQ	= 00000000		
BUF_DESC	0000030A	R	03	FAB\$C_VAR	= 00000002		
BUF_LEN	00000308	R	03	FAB\$L_ALQ	= 00000010		
CCASTHAND	00000B23	R	05	FAB\$L_DEV	= 00000040		
CHAN_BUF	00000312	R	03	FAB\$L_FNA	= 0000002C		
CHECK_IOSB	000008A5	R	05	FAB\$L_FOP	= 00000004		
CHFSL_SIGARGLST	= 00000004			FAB\$L_STS	= 00000008		
CHFSL_SIG_ARG1	= 00000008			FAB\$L_STV	= 0000000C		
CHFSL_SIG_ARGS	= 00000000			FAB\$V_CHAN_MODE	= 00000002		
CHFSL_SIG_NAME	= 00000004			FAB\$V_CR	= 00000001		
CHK_QIO_AST	0000091B	R	05	FAB\$V_FILE_MODE	= 00000004		
CLEAN_EXIT	000007C8	R	05	FAB\$V_GET	= 00000001		
CNTRLMSG	000000A3	R	02	FAB\$V_LNM_MODE	= 00000000		
COMMON	00000AEA	R	05	FAB\$V_PUT	= 00000000		
COMP_STATUS_MSG	000004AF	R	02	FAB\$V_UFO	= 00000011		
CONTROLLER	00000031	R	02	FAB\$V_UPD	= 00000003		
CONT_DESC	000001F5	R	02	FAB\$V_UPI	= 00000006		
CS1	00000082	R	02	FAB\$W_GBC	= 00000048		
CS3	00000094	R	02	FAIL_OUT	00000898	R	05
DDB_RFA	00000BC0	R	03	FAB_BUF	00000004	R	03
DEAD_CTRLNAME	000000E4	R	02	FILE	000001FD	R	02
DEV\$V_TRM	= 00000002			FIND_IT	000001E1	R	05
DEVDEP_SIZE	= 00000000			FLAG	00000002	R	03
DEVDESC	00000218	R	03	FLAG_SHUTDNM	= 00000020		
DEVNAM_LEN	000002E4	R	03	FLAG_SHUTDNV	= 00000005		
DEV_NAME	00000237	R	03	FOUND_IT	00000279	R	05
DIB	00000246	R	03	GOOD_DATA	00000A81	R	03
DIB\$B_DEVCLASS	= 00000004			HALFMIN	000001E5	R	02
DIB\$B_DEVTYPE	= 00000005			ILLEGAL_REC	00000151	R	02
DIB\$K_LENGTH	= 00000074			INADDRESS	000002D2	R	03
DIBBUF	0000024E	R	03	INIDEV_UPDERR	000001B8	R	02
DMF_IOSB_DUMP	00000393	R	02	INI_FAB	00000B2C	R	03
DMP_IOSB_DUMP	00000411	R	02	INI_RAB	00000B7C	R	03
DT\$DMP1T	*****	X	05	INPUT_ITMLST	00000072	R	02
DUMMY_FAB	00000C18	R	03	IOSM_CLR_COUNT	*****	X	05
DUMMY_RAB	00000C68	R	03	IOSM_CTRL	*****	X	05
DVIS_DEVNAM	= 00000020			IOSM_CTRLCAST	*****	X	05
EFN2	= 00000004			IOSM_NOW	*****	X	05

UETI  
V04

20  
6C  
72  
61  
4E

69  
20  
2E

61  
72  
20  
41

66  
69  
61  
44

20  
54

64

41  
66

64  
3A



UETDMPF00  
Symbol table

VAX/VMS UETP DEVICE TEST FOR DMP 11/ DMF 16-SEP-1984 01:24:05 VAX/VMS Macro V04-00  
10-SEP-1984 12:03:55 [UETP.SRC]UETDMPF00.MAR;2

Page 47  
(22)

IOSM_RD_COUNT	*****	X	05	RABSL_STS	= 00000008		
IOSM_SHUTDOWN	*****	X	05	RABSL_STV	= 0000000C		
IOSM_STARTUP	*****	X	05	RABSV_PMT	= 0000001E		
IOS_READVBLK	*****	X	05	RABSW-RFA	= 00000010		
IOS_SENSEMODE	*****	X	05	RABSW-RSZ	= 00000022		
IOS_SETMODE	*****	X	05	RCV_IOSB	00000678	R	03
IOS_WRITEVBLK	*****	X	05	READ_ERRCOUNT	00000778	R	05
ITERATION	000002E6	R	03	READ-PRM	0000031A	R	02
LC_BITM	= 00000020			READ_SIZE	= 00000000		
LIBSSIGNAL	*****	X	05	RECEIVED_MSG	000004E7	R	02
LIMIT	= 00000010			RECORD	00000209	R	02
LOOPBACK_TEST	0000051D	R	05	RECV	00000581	R	05
MAX_DEV_DESIG	= 0000000A			RECVPOOL_SIZ	= 00000004		
MAX_MSG_LEN	= 00000200			RECV_AST	0000093F	R	05
MAX_PROC_NAME	= 0000000F			RECV_BUF	00000880	R	03
MAX_UNIT_DESIG	= 00000005			RECV_EFN	= 00000008		
MODE	00000041	R	02	RECV_ERR_MSG	000004FF	R	02
MODE_IS_ONEM	= 00000010			REC_SIZE	= 00000028		
MODE_IS_ONEV	= 00000004			RESTART	0000043D	R	05
MSG_BLOCK	000002EE	R	03	RMSS_BLN	*****	X	02
NAME_LEN	= 0000000F			RMSS-BUSY	*****	X	02
NEW_NODE	00000A90	R	03	RMSS-CDA	*****	X	02
NMASC_LINCN_LOO	= 00000001			RMSS-FAB	*****	X	02
NMASC_LINPR_POI	= 00000000			RMSS-FACILITY	= 00000001		
NMASC_PCCI_TRI	= 00000474			RMSS-RAB	*****	X	02
NMASC_PCLI_CON	= 00000456			RMS_ERROR	00000AB4	R	05
NMASC_PCLI_PRO	= 00000458			RMS-ERR_STRING	00000217	R	02
NOUNIT_SELECTED	0000012B	R	02	RW_TIME-ID	= 00000003		
NO_CTRNAME	000000C4	R	02	RW-TO MSG	00000275	R	02
NO_RMS_AST_TABLE	0000004D	R	02	SAFE-TO-UPDM	= 00000004		
NRAT_LENGTH	= 00000014			SAFE-TO-UPDV	= 00000002		
OTSSCVT_TI_L	*****	X	05	SEC\$M-EXPREG	*****	X	05
OUTADDRESS-	000002DA	R	03	SEC\$M-GBL	*****	X	05
P1BUF	00000386	R	03	SENSE-ERRMSG	00000541	R	02
P2BUF	0000039E	R	03	SENSE-P1BUF	000003B8	R	03
P2BUF_DESC	00000396	R	03	SENSE-P2BUF	000003C8	R	03
P2BUF_LEN	= 0000000C			SENSE-P2DESC	000003C0	R	03
PAGES	= 00000001			SENSE-P2LEN	= 00000090		
PASS	000002EA	R	03	SENSE-PRM	00000339	R	02
PASS_MSG	00000185	R	02	SENSE-TEST	000005D6	R	05
PMTSIZ	= 00000019			SET-PRM	00000366	R	02
PRM	= 00000064			SET-XMIT BUF	0000052B	R	05
PROCESS_NAME	00000220	R	03	SHRS-ABENDD	= 000010E0		
PROCESS_NAME_FREE	= 0000000B			SHRS-BEGIND	= 00001038		
PROC_CORT_NAME	0000008B	R	05	SHRS-ENDED	= 00001080		
PROMPT	00000238	R	02	SHRS-OPENIN	= 00001098		
QUAD STATUS	000002CA	R	03	SHRS-TEXT	= 00001130		
RAB\$B-PSZ	= 00000034			SS\$-BADPARAM	= 00000014		
RAB\$B-RAC	= 0000001E			SS\$-BUFFEROVF	= 00000601		
RAB\$C-BID	= 00000001			SS\$-CONTROL C	= 00000651		
RAB\$C-BLN	= 00000044			SS\$-ENDOFFILE	= 00000870		
RAB\$C-RFA	= 00000002			SS\$-NORMAL	= 00000001		
RAB\$C-SEQ	= 00000000			SS\$-NOSUCHSEC	= 00000978		
RAB\$C-CTX	= 00000018			SS\$-SSFAIL	= 0000045C		
RAB\$C-FAB	= 0000003C			SS\$-WASSET	= 00000009		
RAB\$C-PBF	= 00000030			SSERROR	000009D1	R	05
RAB\$C-ROP	= 00000004			SS_SYNCH-EFN	= 00000003		

UET  
V04

73  
62  
69

6E  
64

65  
69

6F  
64  
21  
4C

75  
20  
5F  
21

28  
20  
31

44  
4C  
52  
4C

31  
49  
3D

31  
49  
3D  
50  
4C  
28  
58  
52  
41  
2C



UETDMPF00  
Symbol table

VAX/VMS UETP DEVICE TEST FOR DMP 11/4 DMF 16-SEP-1984 01:24:05 VAX/VMS Macro V04-00  
10-SEP-1984 12:03:55 [UETP.SRC]UETDMPF00.MAR;2

Page 48  
(22)

START_CON?	00000461	R	05
START_CONT_PRM	000002A2	R	02
START_TEST	00000407	R	05
START_TO MSG	00000251	R	02
START_TRI	00000492	R	05
START_TRIB_PRM	000002D0	R	02
STATUS	000002C6	R	03
STR\$UPCASE	*****	X	05
STSSK_ERROR	= 00000002		
STSSK_INFO	= 00000003		
STSSK_SUCCESS	= 00000001		
STSSK_WARNING	= 00000000		
STSSM_INHIB MSG	= 10000000		
STSSS_FAC NO	= 0000000C		
STSSS_SEVERITY	= 00000003		
STSSV_FAC NO	= 00000010		
STSSV_SEVERITY	= 00000000		
SUC_EXIT	00000810	R	05
SUPDEV_GBLSEC	00000020	R	02
SUP_FAB	00000BC8	R	03
SYSS\$ASSIGN	*****	GX	05
SYSS\$CANTIM	*****	GX	05
SYSS\$CONNECT	*****	GX	05
SYSS\$CRMPSC	*****	GX	05
SYSS\$DCLEXH	*****	GX	05
SYSS\$EXIT	*****	GX	05
SYSS\$EXPREG	*****	GX	05
SYSS\$FAO	*****	X	05
SYSS\$GET	*****	GX	05
SYSS\$GETDEV	*****	GX	05
SYSS\$GETDVI	*****	GX	05
SYSS\$GETMSG	*****	GX	05
SYSS\$INPUT	00000061	R	02
SYSS\$MGBLSC	*****	GX	05
SYSS\$OPEN	*****	GX	05
SYSS\$PUTMSG	*****	GX	05
SYSS\$QIO	*****	GX	05
SYSS\$QIOW	*****	GX	05
SYSS\$SETAST	*****	GX	05
SYSS\$SETIMR	*****	GX	05
SYSS\$SETPRN	*****	GX	05
SYSS\$SETSFH	*****	GX	05
SYSS\$TRNLOG	*****	GX	05
SYSS\$UPDATE	*****	GX	05
SYSS\$WAITFR	*****	GX	05
SYSIN_FAB	00000A98	R	03
SYSIN_RAB	00000AE8	R	03
TEST_NAME	0000000F	R	02
TEST_OVERM	= 00000002		
TEST_OVERV	= 00000001		
TEXT_BUFFER	= 000000FA		
THREEMIN	000001DD	R	02
TIME_ERR_OUT	0000099E	R	05
TIME_ID T	= 00000001		
TIME_SUC_OUT	000009C9	R	05
TR_PTBUF	0000038E	R	03
TR_P2BUF	000003B2	R	03

TR_P2BUF_DESC	000003AA	R	03
TR_P2BUF_LEN	= 00000006		
TTCHAN	00000000	R	03
UETDMPF00	00000000	RG	05
UETP	= 00740000		
UETPS_ABENDD	= 007410E0		
UETPS_ABORTC	= 0074832B		
UETPS-BEGIND	= 00741038		
UETPS_DATAER	= 00748010		
UETPS_DENOSU	= 00748333		
UETPS_DEUNUS	= 0074819A		
UETPS_ENEDDD	= 00741080		
UETPS_ERBOXPROC	= 00748020		
UETPS_FACILITY	= 00000074		
UETPS_OPENIN	= 00741098		
UETPS_TEXT	= 00741130		
UETUNTSB_FLAGS	= 0000000B		
UETUNTSB_TYPE	= 00000008		
UETUNTSB_FAB	= 00000110		
UETUNTSB_INDSIZ	= 000001A4		
UETUNTSK_FAB	= 00000110		
UETUNTSK_RAB	= 00000160		
UETUNSM_TESTABLE	= 00000002		
UETUNST_FILSPC	= 00000014		
UETUNSV_TESTABLE	= 00000001		
UETUNSW_SIZE	= 00000009		
UNIT_DESC	000001ED	R	02
UNIT_LIST	00000A88	R	03
UNIT_LOOP	00000C98	R	05
UNIT_NUMBER	000002E2	R	03
UPDATE_FAILED	00000CFD	R	05
WRITE_PRN	000002FD	R	02
WRITE_SIZE	= 00000000		
XD_CHAN	00000306	R	03
XD_IOSB	00000670	R	03
XMSM_CHR_LOOPB	= 00000002		
XMIT	00000556	R	05
XMIT_BUF	00000680	R	03
XMIT_EFN	= 00000005		

UETP  
V04-

43  
4C  
67  
65  
2F  
21  
3D  
20  
20  
21  
3D  
20  
4F  
21  
3D  
20  
52  
21



-----  
! Psect synopsis !  
-----

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
RODATA	000005DC ( 1500.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR NOEXE RD NOWRT NOVEC PAGE
RWDATA	00000CAC ( 3244.)	03 ( 3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC PAGE
\$RMSNAM	00000023 ( 35.)	04 ( 4.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
DMPF	00000D51 ( 3409.)	05 ( 5.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE

-----  
! Performance indicators !  
-----

Phase	Page faults	CPU Time	Elapsed Time
Initialization	40	00:00:00.07	00:00:00.58
Command processing	141	00:00:00.67	00:00:05.85
Pass 1	1114	00:00:30.70	00:01:12.41
Symbol table sort	9	00:00:03.39	00:00:07.81
Pass 2	472	00:00:07.37	00:00:16.21
Symbol table output	39	00:00:00.30	00:00:01.08
Psect synopsis output	4	00:00:00.04	00:00:00.06
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1822	00:00:42.55	00:01:44.02

The working set limit was 2000 pages.  
171259 bytes (335 pages) of virtual memory were used to buffer the intermediate code.  
There were 130 pages of symbol table space allocated to hold 2362 non-local and 39 local symbols.  
1811 source lines were read in Pass 1, producing 41 object records in Pass 2.  
59 pages of virtual memory were used to define 52 macros.

-----  
! Macro library statistics !  
-----

Macro library name	Macros defined
-\$255\$DUA28:[UETP.OBJ]UETP.MLB;1	2
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	1
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	46
TOTALS (all libraries)	49

2653 GETS were required to define 49 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:UETDMPF00/OBJ=OBJ\$:UETDMPF00 MSRC\$:UETDMPF00/UPDATE=(ENH\$:UETDMPF00)+EXECML\$/LIB+LIB\$:UETP/LIB



0410 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0411 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

